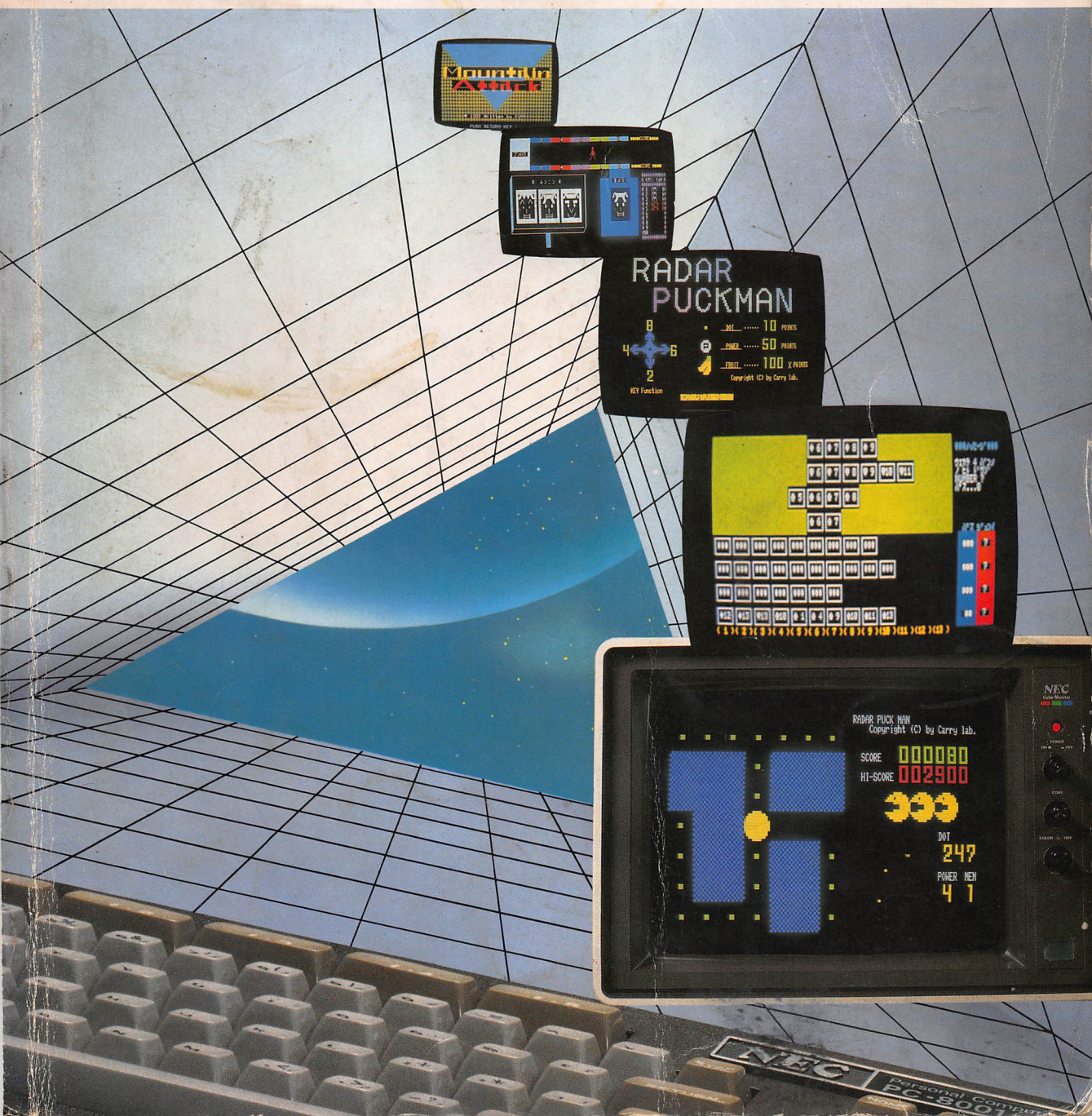
月刊マイコン別冊

PC-8001マシン語入門 II

アセンブラから電子音楽付きカラー・グラフィックまで

MULTIマイコン研究会
塚越 一雄著

9800 ORIGINAL SOFT

作表縦横計算型文字データベース

PARAM/K2 漢字

PC-9800 **69,000円**

- 売上元帳からある得意先や商品だけのデータを引き出し、単独の新ファイルを作成。新ファイルのデータが変わると、同時に元帳のデータ更新も行えます。
- カタカナ3文字で漢字等が8文字まで登録でき、データは記号化や熟語化で入力が容易になりました。
- 1データは127文字＋数値32項目迄指定できます。127文字はユーザーサイドで任意の桁数の分割ができ、項目数を設けられます。
- 項目間の演算や項目の小計、中計、合計、または平均値など、ユーザーサイドの指定式も計算します。
- 入出力は漢字(2965文字)、ひらがな、カタカナ、英数字など、豊富な文字が入出力、プリント使用できます。
- 印刷は項目を組合せることで何種類もの形式が指定でき形式名を指定することで、見やすい漢字で印字します。
- 積層・比較棒グラフを表示・印刷できます。

情報検索型文字データベース

PARAM/K1 漢字

PC-9800 **49,000円**

- 項目(データ名)の数と長さ、画面、プリンター出力が自由指定できます。
- 並べかえ、追加、修正、削除は簡単。
- 1件(1レコード)64文字から127文字まで。
- 複合条件(AND)で検索します。
- 1行は漢字仕様で53文字。
- 複合条件(例えば東京都、男性、25才、未婚)で必要なデータを6秒で検索して表示印刷します。

PARAM/2〔縦横集計型〕

PC-9800 **39,000円**

- 横項目、縦項目の集計は勿論、平均値、小計も算術します。
- 小計グラフ表示も簡単な操作で行います。
- 1件256文字まで。
- 検索・並べかえ・追加・修正・削除機能を持ち操作は簡単です。
- 1行 PC-8023-136桁、EPSON MP-100-233桁。
- 複合条件(例えばTV・12月・5万円以上)で必要なデータを検索して表示印刷します。

PARAM/3〔マトリックスグラフ作成型〕

PC-9800 **39,000円**

- 横の項目、縦の項目を自由に設定できます。
- 横の長さは、NEC PC-8023 プリンターで最大136文字 EPSON MP 100プリンターで最大233文字。
- 縦は200項目設定できます。
- 横項目・縦項目の演算も実行し、必要な場所に自動的に記入します。
- 画面を移動させることによって、必要なデータを即、画面に表示します。
- データの円・棒・折線グラフが即、作成できます。

マイレター98 漢字〔日本語ワードプロセッサー〕

PC-9800〈単語事典20,000語登録〉 **69,000円**

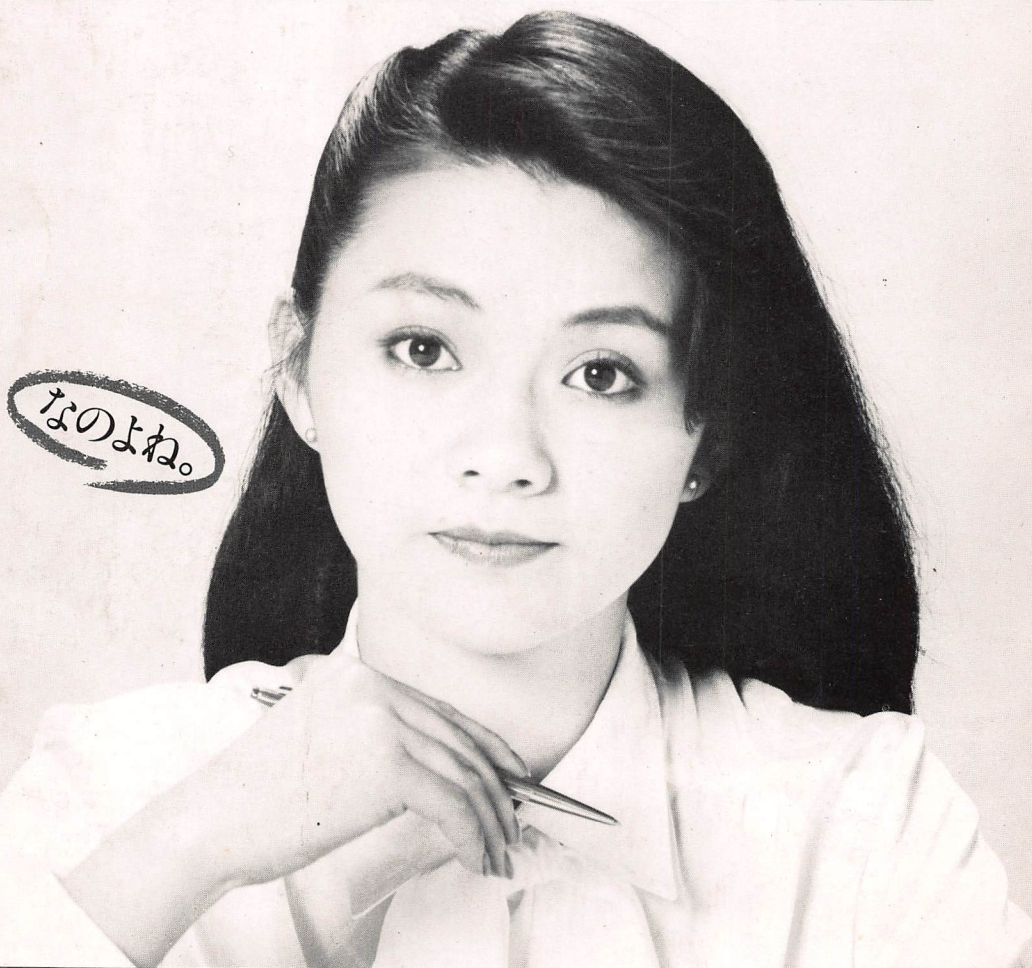
従来のマイレター機能に加えて●1文章＝40字×200行●高速処理●短文登録・熟語登録が可能になるなど、いちだんと仕様が充実しました。 **●縦書が可能●単語事典20,000語登録**

英文ワードプロセッサー〔ワード・メイト〕

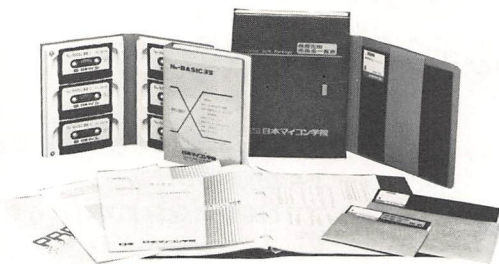
PC-9800 **39,000円**

GRAPH/98〔自動グラフ作成プログラム〕

PC-9800 **19,000円**

16ビットをフルに活用。ソフトの新ステージ。

株式会社 高電社

マイコンショップ・システムイン高電社 〒546 大阪市東住吉区杭全7-10-15 ☎(06)719-1131

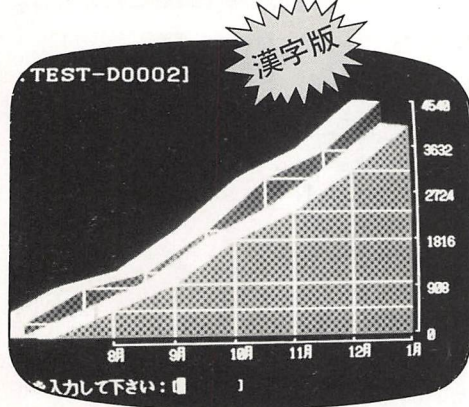
大阪駅前店・システムイン高電社 〒530 大阪市北区梅田1-11-4大阪駅前第4ビル6F ☎(06)341-3371

新製品NEWS

コンサイスシリーズ

ビジネス向け
イージーオペレーションソフト
for PC-9800

■カナ漢入力■漢字仕様■高解度ディスプレイ対応

当シリーズは、ハイグレードなプログラム構成で、一歩進んだビジネスワークを達成していただける秀れた内容。しかもBASICの知識のない方でも簡単に操作していただけるという、大へん扱いやすいソフトです。

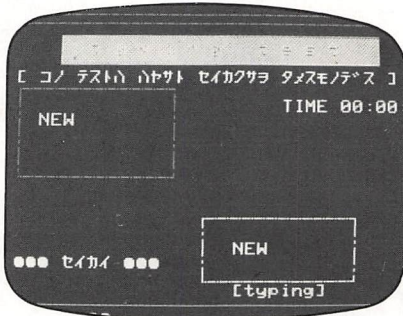
1 得意先別売掛金管理 ディスク版———29,800円

●カナ漢字変換入力により得意先名などを読みやすく漢字で表示できます。●当社が蓄積したノウハウを駆使してマシン語で開発したISFシステムにより、得意先コードをユーザーの思いのままに決定できます。●検索機能を装備しているので、回収率等の条件を設定して、条件に合った得意先をリストアップできます。●グラフ表示で視覚化ができ、回収傾向などが一目で判別できます。●半年分のデータ保管が可能です。

カセット版ソフト紹介——Vol.①

BASIC演習

for PC-8800, PC-9800
カセット6本組 9,800円(ディスク版:20,000円)

当プログラムは、これからBASICの勉強を始めようという方に、画面上でわかりやすく説明したものです。問題も出題されますので、理解できたかどうかのチェックもできます。

〈STEP〉	〈内 容〉
演習Ⅰ	BASICの基礎
演習Ⅱ	画面出力・データ入力・処理制御
演習Ⅲ	グラフィック
演習Ⅳ	ディメンジョン・ソート
演習Ⅴ	プログラミング
演習Ⅵ	タイピング

ビジネス向け
ソフトウェアラインアップ

ディスク版

コンサイス・シリーズ 各29,800円
for PC-9800
——業務用イージーオペレーションソフト——
(カナ漢入力、漢字仕様、高解像ディスプレイ対応)

★得意先別売掛金管理 ★仕入先別買掛金管理
★在庫管理 ★財務(試算表)管理
★予算実績管理
▶カナ漢入力サブルーチン……………9,800円

本格派ビジネスシリーズ
for PC-8001, PC-8800, PC-9800, FM8, PASOPIA(T), if800model30/model20, MULTI16

★顧客管理 ★財務会計
★給与計算 ★グラフパッケージ
★基礎統計 ★多変量解析
★効果検定 ★株価分析
★BASIC演習 ★簡易言語「HOAPS」

カセット版

for PC-8001, PC-8800, FM8, PASOPIA(T), mZ-80B/mZ-2000

★金銭出納帳 ★財務分析
★成長・貢献・分析図(ポートフォリオ)— 以上各5,000円
★簡易言語「日本語ビジック」★得意先別売掛一覧表
★在庫管理 ★予算統制(売上集計)
★資金繰り表 ★損益分岐点算出
★売上計画シミュレーション ★売上分析(Zチャート)
★ABC分析(重点管理)★タイムカード計算
★BASIC演習———— 以上各9,800円
★基礎統計 ……………………28,000円

※ディスク版、カセット版とも、機種によっては一部未開発のソフトもあります。詳細はお問合せ下さい。

PC-8001用ソフト

★アセンブラ(8080)ROMセット………25,000円
★アセンブラ(Z-80)ROMセット………25,000円
★ディスクアセンブラ(Z-80)……………30,000円
★逆アセンブラROM……………………9,800円
★CAP-X(国家試験対策用)ROM… 20,000円

コンバーションプログラム 各20,000円

★PC-8001 ←→ IF800/20
★PC-8001 ——→ マルチ16
★PC-8001 ——→ FM8
★PC-8001 ←→ パソピア(T)
★IF800/20 ——→ FM8
★IF800/20 ——→ マルチ16

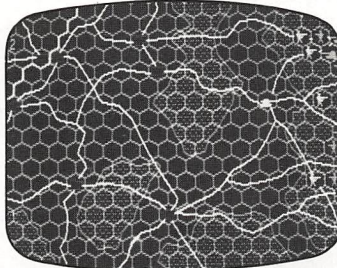
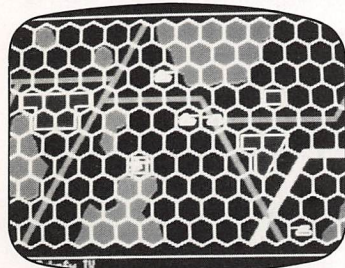
ゲーム ソフト

グラフィック シミュレーションゲーム

シミュレーションゲームにグラフィックがプラスされて、面白さがますますスケールアップしてきた。
〈カセット版〉●PC-8801●PC-9801●X1●FM7/FM8〈ディスク版〉●PC-8801●PC-9801

タンクコンバット

カセット版……………………4,200円
ディスク版……………………6,500円

このゲームは第2次世界大戦末期のヨーロッパ大陸における戦車戦をシミュレートしたものです。あなたはドイツ軍戦車部隊の指揮官。コンピュータが指揮する連合軍戦車部隊と戦ってください。連合軍の司令部を破壊するかすべての戦車を破壊するとあなたの勝ちです。

バルジ大作戦

カセット版……………………4,500円
ディスク版……………………7,000円

第2次世界大戦末期の1944年12月16日、深い霧に包まれたアルデンヌの森にドイツ機甲部隊のエンジン音が響きわたった。ドイツ軍最後の大反攻作戦が開始されたのである。あなたは連合軍の指揮官として、ドイツ軍の攻撃を食い止めることができるだろうか。

クイックアドベンチャーシリーズ（マシン語使用）

新発売

for PC-8001 PC-8001MKII 各3,000円

G-1151 アイスモンスター

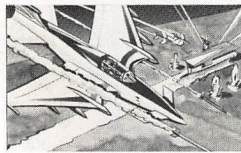
洞窟内に生息するモンスター達。あなたは戦車コロッサスを操縦して、この無数の敵を全滅させてください。

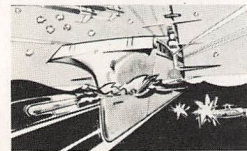
G-1152 メタモルフォーシィス

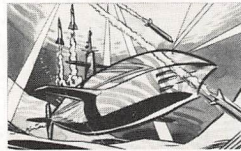
移動、変形をくり返す謎の物体を操作して、敵を打ち殺してください。でも相手は難敵、殺しても残骸が残るのだ。

G-1153 ブラックスネーク

このヘビは獲物を食べるたびにどんどん尾が長くなり、スピードを増してくる。さて、頭をぶつけずにどれだけ長くできるか。

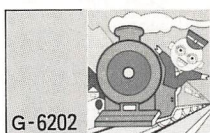
G-1154 スカイシィップ

地球連邦軍の戦闘空母シーウルフは、海底王国のポットと同様に反乱軍のスカイシップにも攻撃を開始した。

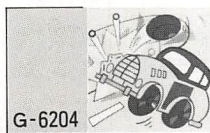
G-1155 シーウルフ

海底王国を探るため太平洋上にいたシーウルフ部隊は、反乱軍のスカイシップ5台に一斉攻撃を浴びせられた。

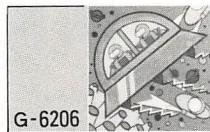
G-1156 ポットメーカー

シーウルフ戦より帰還途上、一隻のポットメーカーが連邦軍のシーウルフと3種の魚雷により攻撃を加えられた。

新発売 2in1シリーズ

各3,000円
for X1, FM7/FM8

A面：ルパン四世
B面：アトランティス

A面：SL
B面：逆襲地下帝国

A面：ザ・ディープ
B面：クレージーストーン

A面：カークラッシュ
B面：サーチャー

A面：灰になるまで
B面：コンクリートブロック

A面：シティ・シューティング
B面：サンド・ウォーズ

サポートセンター ●東京☎03(366)3471 株式会社データマック内 ●姫路☎0792(24)0531 株式会社西松屋内

取扱い販売店募集！

★ご注文は現金書留又は銀行振込をご利用下さい
（取引銀行：三和銀行大阪駅前支店、普通No.82495日本マイコン学院）

信頼は13,000台の保守販売実績から!!　全国送料無料保険付!!

頭金なし「安心全国通販」

イレブン三宮9号店3月5日オープン!!　大阪 ニュー 梅田10号店4月15日オープン!!

☎03-209-5266(代)
コンピュータイレブン
神戸三宮9号店
3月5日オープン!!

頭金なし即決 クレジット1～60回　★超ウルトラ目玉特価セール!!　コンピュータイレブン兄弟会社

日立 レベル-3マークⅡ SET　大特価 21%引　激安

Ⓐ L-3＋高解像カラー＋ソフト29本
定価￥328,800 特価 **￥277,800**
頭金0円　月々4,300円×48回　ボ2万×8回

Ⓑ L-3用 3インチフロッピー 新発売 即納
小さな巨人です **￥79,800**

東芝 PASOPIA 27%引

Ⓐ PASOPIA＋高精細カラー1,600文字
定価￥263,000 特価 **￥198,000**
頭金0円　月々4,700円×24回　ボ3万×4回

Ⓑ PASOPIA＋高解像カラー（640×200）
定価￥283,000 特価 **￥223,000**
頭金0円　月々4,400円×36回　ボ2万×6回

EPSON ハンドヘルドコンピュータ 17%引　ソフト80本

Ⓐ HC-20ハンドヘルドコンピュータ＋ソフト5本
定価￥153,600 **￥128,600**
頭金0円　月々3,600円×30回　ボ1万×5回
頭金0円　月々3,000円×24回　ボ2万×4回

Ⓑ HC-20ハンドヘルドコンピュータ＋マイクロカセット＋ソフト5本
定価￥178,600 **￥148,600**
頭金0円　月々3,400円×36回　ボ1万×6回
頭金0円　月々4,000円×24回　ボ2万×4回

Ⓒ HC-20＋TF-20フロッピー＋￥142,000（ファクトリーオプション除く）
定価￥280,600 **￥230,600**
頭金0円　月々3,000円×36回　ボ3万×6回
頭金0円　月々3,000円×48回　ボ2万×8回

CASIO

Ⓐ FP-1100 激安目玉品 ￥38,200引 30%引
定価￥128,000 ウルトラ 特別価格 **￥89,800**

Ⓑ FP-1100＋高精細カラー＋ソフト5本 ￥100,000引
定価￥263,000 ウルトラ 特別価格 **￥163,000**
頭金0円　月々3,000円×24回　ボ3万×4回　頭金0円　月々4,000円×36回　ボ1万×6回

Ⓒ FP1100＋高解像シャープカラー￥227,800　激安
ウルトラ 特別価格 **￥197,800**　ソフト5本サービス
頭金0円　月々5,400円×48回　ボ　頭金0円　月々3,500円×36回　ボ2万×6回

SHARP パソコンテレビ X1

定価 **￥155,000** カラーモニテレ￥113,000
システム価格 **￥268,000**（ソフト10本サービス）
頭金0円　月々9,300円×36回　頭金0円　月々7,400円×48回
頭金0円　月々4,300円×36回　ボ3万×6回　価格相談

新ソフト紹介 **MZ2000・MZ80B用(ディスク版)**
ビジネス用簡易言語ソフト(データーベース)
今までにない本格的なノンプログラムソフト **￥29,800**

☆周辺機器大特価!!新製品情報提供中!!☆

EPSONプリンター大特価 22%引　プリンタ用紙サービス

Ⓐ 超低価格プリンターRP-80
新発売 **￥89,000**
頭金0円　月々2,700円×24回　ボ1万×4回

Ⓑ 低価格高速プリンターFP-80標準機（F101）
新発売 **￥149,800**
頭金0円　月々3,500円×36回　ボ1万×6回

● FP-80(PC8001用) **￥152,800**
● FP-80(PC8801/9801用) **￥153,800**

新製品 SHARPカラープリンター
10インチ巾8色 **￥228,000**（各メーカー接続可）

Ⓒ MP80K漢字用　￥189,000 プリンタ用紙1,000枚付
￥151,200　頭金0円　月々4,200円×24回　ボ2万×4回

エプソン標準プリンターが ￥4,800(テープ版) 漢字プリンターに変身!! 新漢字ワープロソフト

● GPシリーズ大特価 22%OFF

EPSON TF-20フロッピー

両面倍密(2W)560KB～640KB **￥142,000**

超低価格フロッピー
3インチフロッピーは小さな巨人!!
新発売 日立 レベル3 用
小さな500KB
即納 ￥79,800
超低価格シリーズ

Ⓐ PC8801対応型 ￥142,000＋ファクトリオプション￥24,000＝特価 **￥142,000**
（PC8001対応可）頭金0円　月々3,200円×36回　ボ1万×6回

Ⓑ MICRO-8対応型￥142,000＋ファクトリオプション￥21,000＝特価 **￥139,000**
頭金0円　月々3,100円×36回　ボ1万×6回　頭0円　月々3,800円×48回

Ⓒ MZ80対応型￥142,000＋ファクトリオプション￥38,000＝特価 **￥153,000**
頭金0円　月々3,600円×36回　ボ1万×6回　頭0円　月々4,200円×48回

シャープRGBカラー　超低価格シリーズ　激安

Ⓐ PC9801・FM-11対応|PC8801対応(640×400)
シャープ(コードサービス付) **￥165,000**　限定100台
頭金0円　月々3,100円×24回　ボ3万×4回
頭金0円　月々4,000円×36回　ボ1万×6回

Ⓑ 高解像カラー(640×200) **￥99,800**
シャープ(12インチコード付2,000文字)
頭金0円　月々3,400円×36回　ボ
頭金0円　月々3,200円×24回　ボ1万×4回
ノングレア方式採用

限定100台 Ⓒ 新高解像カラー(2,000文字)　定価 **￥74,800**
頭金0円　月々2,000円×24回　ボ1万×4回
限定300台大特価!!

Ⓓ グリーンモニタ(2,000文字) 特価 **￥19,800**（コードサービス）

ハロー通販
●金利が30%OFFです!!
☎コレクトコール(106番)で!!

テレジット

通販専用申込電話	
東京	☎03(209)5266(代)
名古屋	☎052(451)7374(代)
大阪	☎06(316)0428(代)

☎03-209-5266(代)

☎03-209-5266(代)
受付時間10:00～19:00
コンピュータイレブン兄弟会社
年中無休

PASOCOM PLAZA

東京都新宿区高田馬場 2-17-4菊月ビル5F
ハガキで申し込みの方は必ず☎TEL番号を記入してください。

お支払い方法は現金支払とクレジット支払(ボーナス一括～60回)
〔現金支払〕ハガキ又は電話で連絡の上で現金書留か銀行振込
振込先 富士銀行高田馬場支店 普通 685422 パソコンプラザ
〔クレジット支払〕ハガキ又は電話で連絡した上で手続して下さい。
月々のお支払いは、ご自身の銀行口座から自動引落で ボーナス時払いは(1月と8月です。) 20才以上は保証人 不要です。
銀行口座のない方はお近くの銀行・信金・郵便局よりクレジット会社宛にご送金下さい。(金利10回払8.5%と非常に安い)

パソコンプラザの親会社(株)日本ソフト&ハード社は大学・官庁への納入実績NO.1です。　☎03(209)7376　担当／石川

月刊マイコン82年11月号で記事紹介され話題となった〈書記〉がついにパッケージで登場

日本語ワードプロセッサー

PC-8801

MICRO-8

〈書記〉

© Dempa, System Yoshii

FD不要カセット版

御注文の際は必ず機種名を明記下さい。

定価各9,000円
〒各350円
(カセット1巻，辞書マニュアル1冊　共)

パーソナルコンピュータにおける日本語ワードプロセッサーのソフトパッケージは、最近、色々なものが出回っておりますが、ほとんどフロッピー版である為に、ディスクシステムのコンピュータでしか利用出来ませんでした。このたび、手軽に御利用いただけるカセットテープ版の日本語ワードプロセッサー「書記」を開発いたしました。フロッピー版の日本語ワードプロセッサーに勝るとも劣らないカセットテープベースの決定版です。

①カナ漢字変換、JIS漢字コード入力の両方が使用出来ます。
②拡大文字が使用出来ます。CRTにも拡大表示されます。(MP-80K使用時を除く)
③センタリング、右揃えが出来ます。
④行、文字の挿入、削除が自在に出来ます。
⑤TABセット、改行間隔のピッチ指定が出来ます。
⑥文書をカセットテープへ記録出来ます。

機器構成

《NEC PC-8801》の場合プログラム4種　LOAD "CAS:"
A面、B面同一プログラム

●本体＋漢字ROM　漢字プリンタ　ディスプレイ

●PC-8801 ＋ ●PC-8801-01
- PC-8822
  - PC-8853　A面　前半
  - PC-8053　A面　後半
- MP-80K
  - PC-8853　B面　前半
  - PC-8053　B面　後半

《富士通 MICRO-8 版》の場合プログラム1種　LOAD "CAS:0"
●本体＋漢字ROM＋漢字プリンタ＋ディスプレイ

日本語ワープロ

forMZシリーズ/FD
各49, 800円

© Dempa Carry lab.

対応プリンター ＼ 機種	MZ-2000	MZ-80B
MZ-8P6	JET-2100	JET-1100
MP-130K MP-80K	JET-2000	JET-1000

1 パソコンの基本構成だけで、すぐに日本語ワープロが使えます。

●JET-2000/2100
パーソナルコンピュータ……………………MZ-2000
拡張I/Oポート……………………………MZ-1U01
フロッピーディスク…………………………MZ-80BF
グラフィックボード(ページ1)………MZ-1R01
グラフィックメモリ(ページ2)………MZ-1R02
プリンタ2000……MP-130K、又はMP-80K、
(プリンタインタフェースも含む)
2100……MZ-80P6

●JET-1000/1100
パーソナルコンピュータ……………………MZ-80B
拡張I/Oボックス…………………………MZ-80BK
フロッピーディスク…………………………MZ-80BF
グラフィックRAMI…………………………MZ-80BG
プリンター1000……MP-130K、又はMP-80K、
(プリンタインターフェースも含む)
1100……MZ-80P6

漢字ROM・漢字タブレット不要。JET-2000/1000にはプリンタI/Oカード及び、シャープBASICのリスト出力等を行うソフトウェアをふくむ。MZ-80BP5不可。

2 使用する人に適応する、学習機能付き辞書を装備。

同音異義語は、使用頻度順にならびます。辞書にない熟語は文章を作りながら登録できます。もちろん、慣習用句の略語登録も可能です。時を追うごとに、文章作成はスピードアップしていきます。

3 高度な編集・校正機能をもっています。

BASIC言語のスクリーンエディトを拡張したものですから、特にワープロ用員は不要です。その機能の程は、ぜひワープロ専門機と較べて下さい。

4 B4サイズの文書が一度に作れます。

一頁64文字×64行でその一部(20文字×8行)を画面上に表示します。全体のバランスを見るレイアウト機能や自動作表機能等を有しています。

5 これだけの機能を持って、たったの49,800円です。

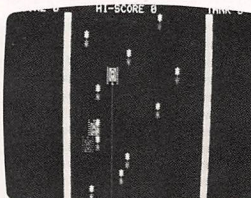
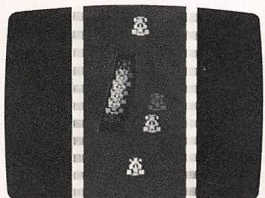
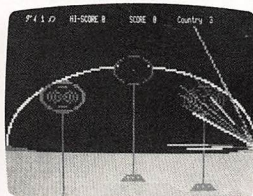
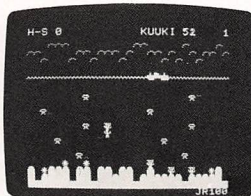
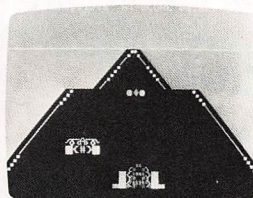
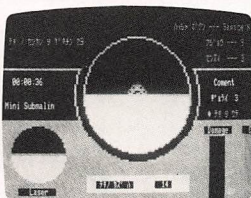
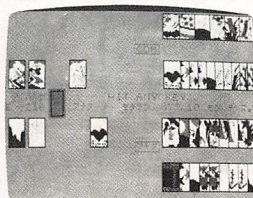
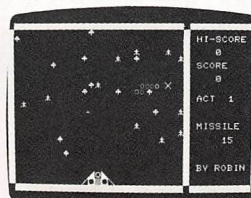
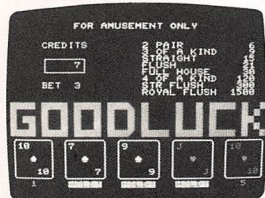
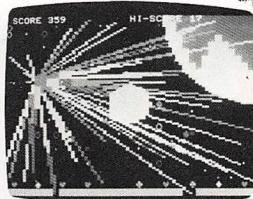
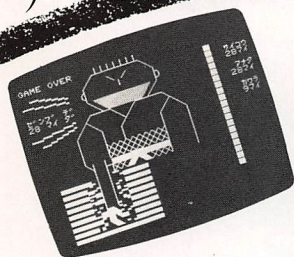
ミニフロッピー1枚、マニュアル。JET-1000/2000では加えて、プリンタI/Oカード、シャープBASIC変更ソフトをふくみます。

電波新聞社出版販売部

御注文は各本社、支局またはマイコンショップ・書店で。

東京本社　〒141 東京都品川区東五反田1-11-15 (03-445-6111)
大阪本社　〒530 大阪市北区中之島3-2-4 (06-203-3361)
西部本社　〒812 福岡市博多区博多駅前2-13-23 (092-431-7411)

Micro Computer PROGRAM CONTEST

マイコン★プログラム★コンテスト

パソコンは、ユーザーが「自由」にプログラミングできるという特徴を持っています。でも、だれもがすぐプログラムを書けるというものでないことも事実です。自分だけのために書いた、あなたのプログラム……。これをうもれさせないで下さい。パソコンユーザーの一人である、あなたが書いたプログラムを求めている、別のユーザーがいるかも知れないのです。

1等1名 10万円

2等5名 5万円

3等10名 3万円

募集内容

ソフトの種類

これまで市販又は発表されていないオリジナルの、パソコンソフト(ゲーム、実務)。

対象機種

一般にパーソナルコンピュータと呼ばれる機種であれば何でも可。

応募方法

プログラム内容、使用方法をくわしく説明した文章とカセットテープ、又はディスケット(使用システムを記入して下さい)。

締め切り

第1回目：昭和58年3月末日

選考・発表

昭和58年4月、月刊マイコン5月号誌上。尚、入選者には別個直接ご連絡いたします。

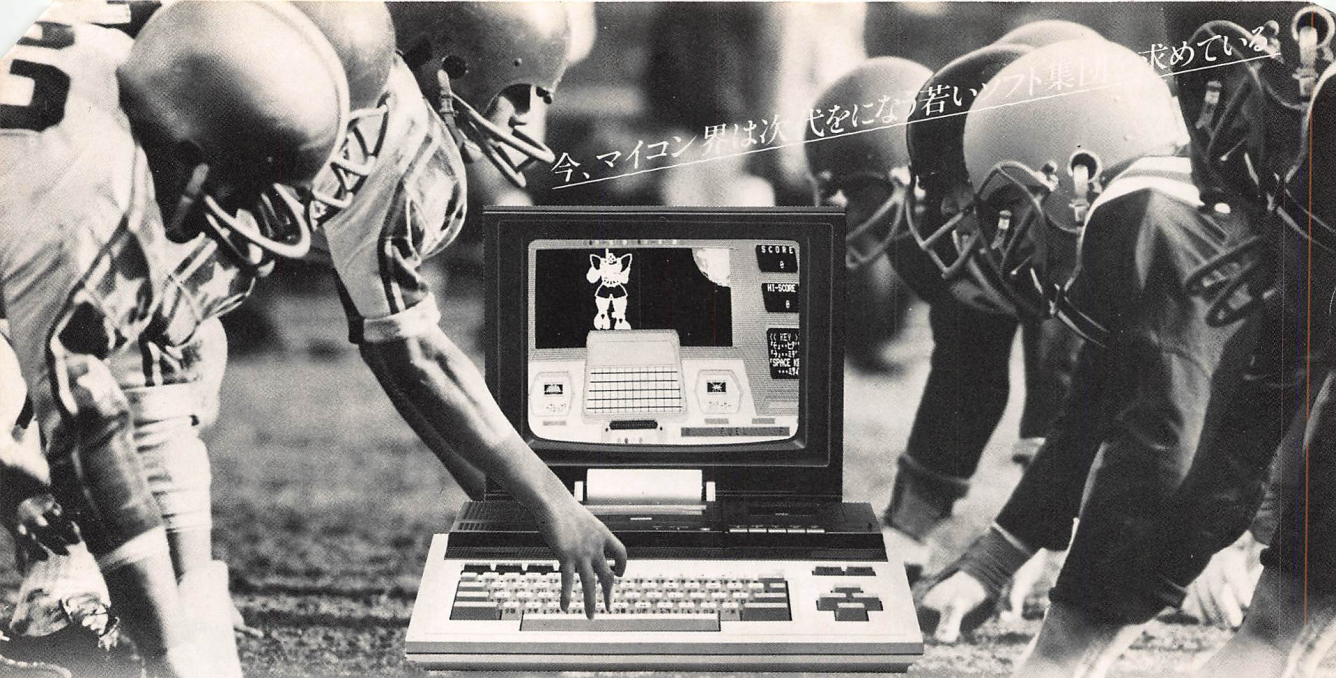
※応募いただきました作品は返却できませんのでご了承下さい。又、入選作品の権利は当社に属するものといたします。

送り先

〒105 東京都港区東新橋1−1−16 東タイビル内

S.T.I.科学技術情報㈱

TEL：03(445)1561

今、マイコン界は次代をになう若いソフト集団を求めている

月刊マイコン 大学クラブ対抗

BEST PROGRAM CONTEST
ベスト・プログラムコンテスト

今、プログラムを書くことについて、専用プログラマーと対等に活躍しているのが若い大学生層であると言っても過言ではありません。月刊マイコン大学クラブ対抗ベストプログラムコンテストは、月刊マイコン誌上で大学生のみなさんの自由な技術発表、情報交換を図り、学生層のプログラム技術の向上と創造的応用の拡大の場を作ることを目的として実施されるものです。各大学のマイコン関連のクラブ、サークル、グループ自信の作品をお送り下さい。毎月優秀クラブを表彰するとともに、クラブを紹介、作品についても誌上にて掲載、紹介してまいります。

応募規定

部門

総合プログラム部門
マシンの性能をフルに引きだすプログラム、アイデアあふれるプログラム等。
使用言語はマシン語及びBASIC。
内容は自由。

エンターテイメント部門
やり始めたら止められなくなるような、おもしろいゲーム。

対象機種 パーソナルコンピュータ全般

作品内容 ………………………………
①プログラム内容，機能，使用方法等の説明文。フローチャート及びプログラミング上の特異点も含む。
②プログラムリスト(できればプリンタ出力で)
③プログラムを納めたカセット又はディスケット。
④あなたのクラブ(サークル，グループ等)の活動内容等を含めた自己紹介。

応募締切 毎月15日

選考 ………………………………
各界コンピュータ専門家，マイコン編集部等で構成する選考委員により実施。

発表 ………………………………
月刊マイコン誌上にて毎月継続的に発表。入賞作品は，入賞クラブ紹介とともに誌上にて掲載してまいります。
※応募作品は，特定企業の権利に属さぬ未発表の創作になるものとします。

賞金 ***BEST 1***…20万円
BEST 2…10万円
BEST 3… 5万円

副賞 ベスト1～3の応募クラブに対してそれぞれ豪華楯を贈ります。

応募先 ………………………………
〒141 東京都品川区東五反田1-11-15 電波新聞社 出版部
月刊マイコン大学クラブ対抗コンテスト係

はじめに

お元気ですか？
長らくお待たせいたしましたが，ここに
PC－8001
マシン語入門（第二巻）
が完成し，ようやくお届けできることになりました。その間，多くの方からたくさんの励ましのお便りをいただき，有難うございました。いま，目の前にある高さ10cmにも及ぶ原稿用紙の山に，感無量の思いにひたっています。

第二巻では，引続き
実験を通し，
実際に**自分の目で確かめながら**
学習を進めて行きます。そして，最終目標を
電子音楽付き
カラー・グラフィックの実現
に置いています。長い１冊ですが，より楽しいマイコンライフを目差して頑張ってください。

なお，多くの独習者の便宜をはかるため
- **紙面に糸目をつけない説明**
- **豊富な図**
- **トリッキーな構成**

のスタイルはくずしていません。また検索の便を高めるため，**目次を詳しく**してあります。第一巻同様，安心してあなたのPC-8001で
マシン語の世界
をお楽しみください。

1983年１月12日
MULTIマイコン研究会

塚　越　一　雄

もくじ

第1ブロック　アセンブラに挑戦

第1章　コメントとORG

第2章　ラベルの威力

第3章　文字列出力ルーチンをめぐって

第4章　中間言語を見る

第5章　ワーク・エリアの設定

第6章　最後はEND

第2ブロック　USR関数への招待

第7章　ミニ・レジスタ表示プログラム

第8章　リロケータブルの世界

第9章　USR関数

第3ブロック　浮動小数点型式とストリング・デスクリプタを探る

第10章　浮動小数点アキュームレータ

もくじ

第11章　BASIC＋マシン語とのリンクの実際

第12章　浮動小数点型式とストリング・デスクリプタ

第4ブロック　効果音付きカラー・グラフィックの世界

第13章　カラー・グラフィックの世界

第14章　音楽演奏に挑戦

付録

第1ブロック

アセンブラに挑戦

宇宙船はユニバーサル映画"E.T." THE EXTRA-TERRESTRIALより

〈はじめに〉

マシン語に出会い，マシン語に触れ，その面白さがわかった時，あなたの

マシン語への旅

が始まります。それは，終りのない長い長い旅かもしれません。そして，その一つの節として

オール・マシン語版

スペース・インベーダー（©タイトー）

の製作が待っています。

あなたがマシン語に慣れた頃，

ハンド・アセンブル

の面白さを知るでしょう。と同時に，その**不便さ**を痛感するかもしれません。

長い，長いマシン語のプログラムを作るとき，ハンド・アセンブルによるプログラミングを手助けしてくれる便利な道具があります。それが，

アセンブラ

です。

アセンブラは，**アセンブリ言語をマシン語に変換**してくれるプログラムです。したがって，それを使用する者は，

マシン語の使用者

です。この前提があるため，アセンブラのマニュアルは，BASICのそれに比べ，

簡素化＝不親切

であるかもしれません。

「ＰＣ-8001　マシン語入門」（第二巻），最初の挑戦は，その

アセンブラ

にアタックします。そうです。この**第１ブロック**では，やがてあなたがアセンブラに接するとき困らないように，

アセンブラ特有の命令

にチャレンジしていくことになります。例によって，そこにはたくさんの

実験例

が用意されています。その中で，**マシン語をやっていく上での，有益なヒント**にも出会うことでしょう。

ＯＫ？　本ブロックは，以後のブロックに進む上で避けては通れぬ

通行手形の交付を受ける

重要なブロックなのです。

なお，現在アセンブラをお持ちの方は，本ブロックで展開される種々の実験を，ぜひあなたの

アセンブラを通して

実施してみてください。それは，あなたにより深**い理解**を与えてくれるでしょう。と同時に，あなたのアセンブラに対する愛着が，より深いものになることでしょう。

では，第一ブロック，ソロリ，ソロリとまいりましょう。

第1章

コメントとORG

```
D000: CD 93 DC CD 9F DC CD C3 DC CD FF DC CD 13 D2 3A ; 0B84
D010: AA E3 3D 28 F4 3D 28 EE CD B2 DD 18 E6 00 00 00 ; 0793
D020: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; 0000
D030: 00 00 CD E2 DB CD 6C D2 CD A0 D2 CD C3 D2 CD F6 ; 0AF6
D040: D2 CD 1D D3 CD 72 D3 CD A7 D3 CD D4 D3 06 14 CD ; 0A43
D050: 3D D4 21 13 16 22 AD E3 06 02 CD 58 D4 CD 74 D4 ; 0723
D060: 06 30 CD 3D D4 3E 32 32 AE E3 06 02 CD 58 D4 CD ; 0715
D070: CB D0 CD ED D0 CD FD D0 CD ED D0 CD 10 D1 CD ED ; 0CAE
D080: D0 CD 2E D1 CD ED D0 CD 67 D1 CD ED D0 CD 7C D1 ; 0BCF
D090: CD ED D0 CD E6 D1 CD ED D0 CD 91 D1 3E 03 32 AB ; 0AE5
D0A0: E3 CD CC D1 21 13 11 22 AD E3 06 03 CD 58 D4 06 ; 074C
D0B0: 3C CD 3D D4 21 13 3E 22 AD E3 06 03 CD 58 D4 21 ; 0661
D0C0: AB E3 35 20 DC C3 32 D0 25 25 2D CD 20 D7 AF 01 ; 076F
D0D0: 02 09 21 01 2E CD 4E D9 21 50 BB 22 CA F3 22 42 ; 05BB
D0E0: F4 01 02 15 11 9F E2 21 01 26 C3 9C DD 0E 0A 06 ; 0540
D0F0: 32 CD 85 D9 CD 8C D2 10 F8 0D 20 F3 C9 3E 68 2E ; 084D
D100: 03 CD 60 D9 21 03 20 CD 0C D9 11 C9 E2 C3 27 D9 ; 077E
D110: 11 F1 E2 21 A9 F5 CD 1D D9 21 05 14 22 CD E3 CD ; 083F
D120: 1F DB 21 06 28 CD 0C D9 11 FB E2 C3 27 D9 11 09 ; 06C6
D130: E3 2E 07 1A A7 28 1B D5 06 02 E5 C5 F5 CD 14 D9 ; 0752
D140: F1 C1 23 77 23 36 28 23 36 E8 E1 2C 10 EC D1 13 ; 06FB
D150: 18 E1 11 AD DF 21 07 14 CD 72 D5 21 08 28 CD 0C ; 0610
D160: D9 11 0D E3 C3 27 D9 11 C5 DF 21 09 14 CD 72 D5 ; 07A4
D170: 21 0A 28 CD 0C D9 11 15 E3 C3 27 D9 11 DD DF 21 ; 06BF
```

そろり・そろりと　まいろうぞ……

恐るべき進歩

お元気ですか？　ごぶさたしています。再びまたあなたの

目・頭・手

を汚す時がやってきました。鈴木さん，田中さん，佐藤さん，一郎君に，次郎君に，太郎さん，おひさしぶりです。さあ，我々の

マシン語への挑戦！

その二回目の旅に，これから出発することに致しましょう！　ハイです。

といったところで，さっそく**第1図**を御覧ください。

「ヤヤ，長いアセンブラのリスト！」

と，まあ驚かないでください。それよりも**恐るべきことは**，今，我々はこの図を見て

〝アセンブラのリスト〞

だとわかることの方が驚異だと思いませんか？

思わない人は，「PC-8001　マシン語入門」の第一巻を開いてみてください。そしてP.16の**第1-1図**を御覧ください。我々が初めてマシン語の勉強を始めた頃，その図の意味はわからなかったのですよ。それが今，何とあなたは，それよりも高級な第1図を見て，

〝アセンブラのリストだ！〞

とわかるようになったのです。そればかりではありません。**第1図**のアセンブラ・リストから，あなたのPC-8001に**マシン語を入力する**ことも，**走らせる**こともできるようになっているのです。つい何ヶ月（いや何年，いや何日）前のあなたと比べてみてください。ね？　**驚異**でしょう？

これもひとえに，あなたが頑張って，「PC-8001マシン語入門」第一巻を読破したからです。そしてチョッピリ，この本を他書に類を見ない廉価で提供してくれた電波新聞社に感謝致しましょう——エヘヘ。

アセンブラ・リストからマシン語を

さて，第一巻では，おもに

マシン語のリスト

```
                    ;====================================
                    ;  MINI REGISTER DISPLAY
                    ;    (81年 3月 18日):BY K.TSUKAGOSHI
                    ;====================================
                    ;
                             ORG   0D000H
                    ;
0257                OCRT:    EQU   257H           ;OUT CRT
5C66                GMON:    EQU   5C66H          ;GOTO MONITOR
5EC0                PRHL:    EQU   5EC0H          ;PRINT HL
5FCA                CRLF:    EQU   5FCAH
5FD4                PSPC:    EQU   5FD4H          ;PRINT SPACE
                    ;
D000 FDE5           MAIN:    PUSH IY              ;REGISTER STORE FOR DISPLAY
D002 DDE5                    PUSH IX
D004 E5                      PUSH HL
D005 D5                      PUSH DE
D006 C5                      PUSH BC
D007 F5                      PUSH AF
                    ;
D008 CDCA5F                  CALL CRLF            ;CARRIAGE LINE FEED,PREPRA FOR DISPLAY
                    ;
D00B 0625                    LD   B,37            ;'AF-SP'=37 CHARACTER
D00D 210000                  LD   HL,0            ;LET HL=SP
D010 39                      ADD  HL,SP
D011 2B                      DEC  HL
D012 2B                      DEC  HL
D013 F9                      LD   SP,HL
D014 E1                      POP  HL
D015 113500                  LD   DE,53           ;HL=ADR(DATA)
D018 19                      ADD  HL,DE
                    ;
D019 7E             MA1:     LD   A,(HL)          ;PRINT 'AF-SP'
D01A CD5702                  CALL OCRT
D01D 23                      INC  HL
D01E 10F9                    DJNZ MA1
                    ;
D020 CDCA5F                  CALL CRLF
                    ;
D023 0606                    LD   B,6             ;6 REGISTERS
D025 E1             MA2:     POP  HL
D026 CDC05E                  CALL PRHL
D029 CDD45F                  CALL PSPC
D02C 10F7                    DJNZ MA2
                    ;
D02E E1                      POP  HL              ;PRINT PC
D02F 2B                      DEC  HL
D030 CDC05E                  CALL PRHL
D033 CDD45F                  CALL PSPC
                    ;
D036 210000                  LD   HL,0            ;PRINT SP
D039 39                      ADD  HL,SP
D03A CDC05E                  CALL PRHL
                    ;
D03D C3665C                  JP   GMON            ;GOTO MONITOR
                    ;
D040 41462020       DATA:    DB   'AF   BC   DE   HL   '
D044 20424320
D048 20204445
D04C 20202048
D050 4C202020

D054 49582020                DB   'IX   IY   PC   SP'
D058 20495920
D05C 20205043
D060 20202053
D064 50
                    ;
D065                         END
```

第1図　長いアセンブラのリスト

で話しを進めてまいりました。しかし前節でも確認しましたように，すでに我々は

マシン語の中級者？

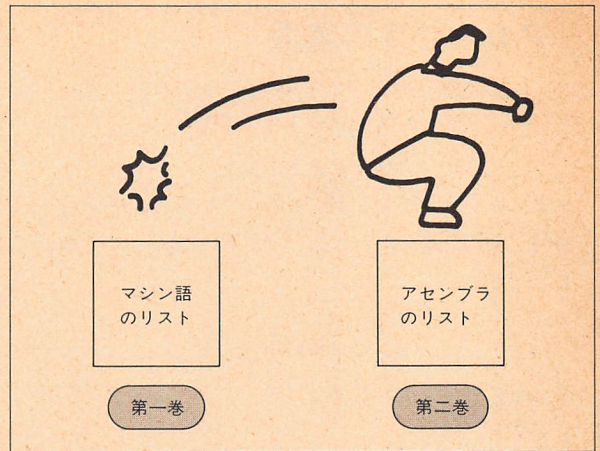
に格が上がっています。そこでと言っては何ですが，第二巻では，もう少し高級な

アセンブラのリスト

で話しを進めていきたいと考えています（**第2図**）。

これは，将来，あなたがマシン語のプログラムを組む際，おそらく

アセンブラ

を利用するであろう，と考えたからです。そのときのため，少しでもアセンブラのリストに慣れていただこうというわけです。

第2図　マシン語からアセンブラへ

さあ，そこで第1図です。あなたは，このリストを御覧になって，ただちにその**マシン語をあなたのPC－8001に打ち込めますね？**

電源ON

MON↘

（本書では↘をRETキーの略記号として使います。）

で**マシン語のコマンド・レベル**にして，

Sコマンドで入力

しようと――，おっと，マシン語の**スタート番地**は？OK，D000Hですね。

SD000↘

これで入力開始です（**第3図**）。また，どの部分を入力すれば良いかは，**第4図**を御覧ください。

```
NEC PC-8001 BASIC Ver 1.2
Copyright 1979 (C) by Microsoft

Ok
mon
*SD000
D000 00-FD 00-E5 00-DD 00-E5
D004 00-E5 00-D5 00-
```


第3図　マシン語を入力する

```
                   ;====================================
                   ;  MINI REGISTER DISPLAY
                   ;    (81年 3月 18日):BY K.TSUKAGOSHI
                   ;====================================
                   ;
                          ORG   0D000H
                   ;
0257               OCRT:  EQU   257H          ;OUT CRT
5C66               GMON:  EQU   5C66H         ;GOTO MONITOR
5EC0               PRHL:  EQU   5EC0H         ;PRINT HL
5FCA               CRLF:  EQU   5FCAH
5FD4               PSPC:  EQU   5FD4H         ;PRINT SPACE
                   ;
D000 FDE5          MAIN:  PUSH  IY            ;REGISTER STORE FOR DISPLAY
D002 DDE5                 PUSH  IX
D004 E5                   PUSH  HL
D005 D5                   PUSH  DE
D006 C5                   PUSH  BC
D007 F5                   PUSH  AF
                   ;
D008 CDCA5F               CALL  CRLF          ;CARRIAGE LINE FEED,PREPRA FOR DISPLAY
```


第4図　どの部分をキーインする？

アセンブラの定義

以上までの手順を，すべての読者の皆さんに確認していただくため，ここで

アセンブラ・リストの見方

をまとめておきましょう。これについては，「PC－8001　マシン語入門」の第一巻，第4章以下でも触れましたね。

ひとくちに〝**アセンブラ**〟といっても，それこそ千差万別です。JISによるアセンブラの定義を御紹介しますと，

> **〈アセンブラ言語〉** assembly language
> 計算機向け言語の一つであって，その命令が通常，計算機命令と1対1対応しており，マクロ命令の使用のような機能を備えうるもの。
>
> **〈アセンブルする〉** to assemble
> アセンブラ言語で表されたプログラムを計算機言語に翻訳することと，多くの場合さらにサブルーチンを連係すること。
> (注)　アセンブルは，通常，アセンブラ言語の演算コードを計算機言語の演算コードに置き換え，記号アドレスを絶対アドレス，即値アドレス，再配置可能アドレスまたは仮想アドレスに置き換えることによって達成される。
>
> **〈アセンブラ〉** assembler
> アセンブルするために使われる計算機プログラム

コメントにもいろいろありまして

話しをPC－8001に限っても，現在，さまざまなタイプのアセンブラが発表・発売されています。**供給デバイス**で見れば，

カセット・バージョン
ディスク・バージョン
ROMタイプ

また，使用する**ニーモニック**で見れば，

インテル・タイプ
ザイログ・タイプ

発生する**マシン・コード**で見れば，

8080オンリー
Z—80の準拠

等々，さまざまですね。値段もタダから，数万円まで。

アセンブラの違いをゴタゴタ並べていても話しが始まりません。ここでは，話しを

アセンブラ・リストの見方

に絞り，一般的なものだけに限ってみます。

まず**コメント**について。

コメントは，BASICの**REM**と同じく**注釈**のことです。コメントは，

プログラム・リストを見やすくする

ために，また後日の

メンテナンスをしやすくするため

プログラムの適材適所に挿入します。

コメントには，2種類のものがあります。すなわち

① **行全体がコメント**となるもの
② 行の途中，**アセンブリ言語の後におく**もの

の2種類です。

①については，**文の1字目**が

C
*
;

のときその**行全体を注釈行と見なす**等いろいろなものがあります。またそれらの併用を認めるものもあります。本書に示すアセンブラ・リストは，

最初が；で始まる行は，その行全体はコメントである

ものと解釈してください(**第5図**)。

次に②について。

文の途中からコメントを置く場合，

文の途中に；が現われると，それ以降はコメントと見なす —Ⓐ

もの。また中には，

アセンブリ言語の後，1字スペースを置けばそこからコメントを書いて良い

といったものもあります。

本書に示すアセンブラ・リストは，**Ⓐのタイプ**とします。**第6図**で御確認ください。

たとえばこの行は，１字目が；なので注釈行となる。
したがって，この行についてはマシン語が発生しない。

ここも同じ

```
                ;====================================
                ;  MINI REGISTER DISPLAY
                ;     (81年 3月 18日):BY K.TSUKAGOSHI
                ;====================================
                ;
                        ORG   0D000H
                ;
0257            OCRT:   EQU   257H          ;OUT CRT
5C66            GMON:   EQU   5C66H         ;GOTO MONITOR
5EC0            PRHL:   EQU   5EC0H         ;PRINT HL
5FCA            CRLF:   EQU   5FCAH
5FD4            PSPC:   EQU   5FD4H         ;PRINT SPACE
                ;
D000 FDE5       MAIN:   PUSH IY             ;REGISTER STORE FOR DISPLAY
D002 DDE5               PUSH IX
D004 E5                 PUSH HL
D005 D5                 PUSH DE
D006 C5                 PUSH BC
D007 F5                 PUSH AF
                ;
```

注釈のない　注釈行。プログラム単位毎に読みやすくする目的で入れてある。

第５図　文全体が注釈行となる

```
                ;====================================
                ;  MINI REGISTER DISPLAY
                ;     (81年 3月 18日):BY K.TSUKAGOSHI
                ;====================================
                ;
                        ORG   0D000H
                ;
0257            OCRT:   EQU   257H          ;OUT CRT
5C66            GMON:   EQU   5C66H         ;GOTO MONITOR
5EC0            PRHL:   EQU   5EC0H         ;PRINT HL
5FCA            CRLF:   EQU   5FCAH
5FD4            PSPC:   EQU   5FD4H         ;PRINT SPACE
                ;
D000 FDE5       MAIN:   PUSH IY             ;REGISTER STORE FOR DISPLAY
D002 DDE5               PUSH IX
D004 E5                 PUSH HL
D005 D5                 PUSH DE
D006 C5                 PUSH BC
D007 F5                 PUSH AF
```

第６図　行の途中からコメントを入れる

複数のＯＲＧ

次に第１図に戻りまして，６行目を御覧ください。

　　ＯＲＧ

とあります。**ＯＲＧ**は，

　　origin〔*ɔ́ridʒin*〕

　　　n. 発端，起源

の略で，感覚的には

　　プログラムのスタート番地

と解釈してください。たとえば，第１図では

　　ＯＲＧ　０Ｄ０００Ｈ

となっています。したがって

　　スタート番地＝Ｄ０００Ｈ

です。以下のマシン語を見ても，そうなっているのがわかりますね？

ところで，ＯＲＧは

　　一つのプログラム中，複数個置ける

というのは御存知ですか？　こうなると

　　ＯＲＧ＝スタート番地

という解釈では感心しません。たとえば，

　　ＯＲＧ　８０００Ｈ

ORG　9000H

∫　　　　　　　　　　　　　　］──Ⓑ

のように，ORGを同一プログラム中に二ヶ所使うと，

Ⓐの部分：8000Hから

Ⓑの部分：9000Hから

のようにアセンブルされます。さらに理解を確実にするため，**第7図**をご覧ください。これは，第1図のリストの途中に，ORGを二つ追加してみました。

Ⓐの部分：D000Hから

Ⓑの部分：E000Hから

Ⓒの部分：8000Hから

アセンブルされているのがお分かりになると思います。

ORGについてまとめると，次のようになります。

① プログラムの最初には（コメント行を除く），必ずORGを置く。この場合は，

ORG＝プログラムのスタート番地

になります。

② プログラムの途中から，アセンブルする番地を変えたい時は，ORGを置くことができる。その場合，アセンブラはそこから

ORGで指定した番地

でアセンブルしていく。

```
                    ;====================================
                    ;  MINI REGISTER DISPLAY
                    ;    (81年 3月 18日):BY K.TSUKAGOSHI
                    ;====================================
                    ;
                            ORG   0D000H
                    ;
 0257               OCRT:   EQU   257H          ;OUT CRT
 5C66               GMON:   EQU   5C66H         ;GOTO MONITOR
 5EC0               PRHL:   EQU   5EC0H         ;PRINT HL
 5FCA               CRLF:   EQU   5FCAH
 5FD4               PSPC:   EQU   5FD4H         ;PRINT SPACE
                    ;
 D000 FDE5          MAIN:   PUSH IY             ;REGISTER STORE FOR DISPLAY
 D002 DDE5                  PUSH IX
 D004 E5                    PUSH HL
 D005 D5                    PUSH DE
 D006 C5                    PUSH BC
 D007 F5                    PUSH AF
                    ;
                            ORG   0E000H
                    ;
 E000 CDCA5F                CALL CRLF           ;CARRIAGE LINE FEED,PREPRA FOR DISPLAY
                    ;
 E003 0625                  LD    B,37          ;'AF-SP'=37 CHARACTER
 E005 210000                LD    HL,0          ;LET HL=SP
 E008 39                    ADD   HL,SP
 E009 2B                    DEC   HL
 E00A 2B                    DEC   HL
 E00B F9                    LD    SP,HL
 E00C E1                    POP   HL
 E00D 113500                LD    DE,53         ;HL=ADR(DATA)
 E010 19                    ADD   HL,DE
                    ;
                            ORG   8000H
                    ;
 8000 7E            MA1:    LD    A,(HL)        ;PRINT 'AF-SP'
 8001 CD5702                CALL OCRT
 8004 23                    INC   HL
 8005 10F9                  DJNZ MA1
                    ;
 8007 CDCA5F                CALL CRLF
                    ;
```


第7図　ORG命令を複数個置く

第2章

ラベルの威力

```
D180: 0B 14 CD 72 D5 21 0C 28 CD 0C D9 11 1D E3 C3 27 ; 0635
D190: D9 3E E8 32 29 FD 21 40 EB 22 32 FD 2E 13 CD 14 ; 0713
D1A0: D9 23 11 55 E3 CD 27 D9 21 15 05 CD 0C D9 11 25 ; 0635
D1B0: E3 CD 27 D9 21 15 41 CD 0C D9 11 2F E3 CD 27 D9 ; 07C9
D1C0: 21 13 1A CD 0C D9 11 39 E3 C3 27 D9 CD 12 D4 21 ; 06C4
D1D0: AC E3 35 CD F3 D3 06 14 CD 73 D9 CD 8C D2 3A AC ; 099B
D1E0: E3 FE 0F 20 E7 C9 06 02 2E 0D C5 E5 CD 14 D9 23 ; 078A
D1F0: 11 5B E3 CD 27 D9 E1 C1 2C 10 EF 21 0D 14 CD 0C ; 0704
D200: D9 11 66 E3 CD 27 D9 21 0E 14 CD 0C D9 11 90 E3 ; 0779
D210: C3 27 D9 3A AA E3 A7 C0 CD F1 0C DC 66 D2 3A B0 ; 09B9
D220: E3 47 3A AB E3 A0 CC DE D4 3A AB E3 E6 0F CC CB ; 0A64
D230: DA 3A B0 E3 47 3A AB E3 A0 CC 3E DE 3A AB E3 E6 ; 09EC
D240: 07 CC 36 DB 3A AB E3 E6 03 CC 7B DB 3A AB E3 E6 ; 0965
D250: 0F CC C8 DE CD B2 D7 CD 19 D8 21 AB E3 34 CD 1D ; 0962
D260: DD CD 85 D9 18 AD CD F1 0C 30 FB C9 21 19 FE 11 ; 08D4
D270: DB E1 CD 27 D9 21 91 FE 13 CD 27 D9 21 CE FD 11 ; 0916
D280: E7 E1 CD 27 D9 21 5E FE 13 C3 27 D9 DB 09 FE FE ; 09C8
D290: CA 66 5C FE DF C0 F1 F1 C9 00 00 00 00 00 00 00 ; 06D4
D2A0: 21 C9 F3 11 59 E2 CD 27 D9 21 41 F4 11 59 E2 CD ; 0865
D2B0: 27 D9 3E 58 2E 03 06 06 C5 E5 CD 60 D9 E1 C1 2C ; 0751
D2C0: 10 F6 C9 11 DD DF DD 21 60 E2 AF 32 B9 E3 DD 7E ; 09B4
D2D0: 00 A7 C8 D5 67 DD 7E 01 6F E5 CD 72 D5 DD 23 DD ; 094C
D2E0: 23 21 B9 E3 7E 2F 77 06 50 CD 73 D9 E1 CD 6B D5 ; 0861
D2F0: D1 CD 8C D2 18 D8 11 DD DF 21 01 2E 7C FE 4C C8 ; 0897
```

なまえ　と　らべる　と……

方法が異なる

アセンブラ・リストの見方について，次の説明に移る前に，一つ**簡単なプログラム**を見てみることにします。

第8図のプログラムを御覧ください。

非常に短いマシン語のプログラムです。

　　C100H～C107H

から成る**わずか8バイト**のプログラムです。これは，いったい何をするプログラムでしょう？

さっそく**入力**してみましょう。

　MON↘

　SC100↘

　3E

　E9

　　⁝

簡単に終了しますね(**第9図**)？　続いてDコマンドで**入力の確認**をしましょう（**第10図**)。続いて

```
                  ;==================
                  ;  PRINT ♥
                  ;    (CALL 257H)
                  ;==================
                  ;
                          ORG   0C100H
                  ;
C100 3EE9                 LD    A,0E9H          ;LET A='♥'
C102 CD5702               CALL  257H
C105 C3665C               JP    5C66H           ;GOTO MONITOR
                  ;
                          END
```

第8図　何のプログラムでしょう？

```
*SC100
C100 C2-3E 01-E9 0D-CD 25-57 CD-02 49-C3 C0-66 38-5C
C108 06-
*
```

第9図　Sコマンドで第8図のプログラムを入力する

```
*DC100,C107
C100 3E E9 CD 57 02 C3 66 5C
*
```

第10図　Dコマンドで入力の確認をする

ＧＣ１００↘
で**プログラムの実行**です（第11図）。

プログラムを実行させると，
　　　♥
が一つ表示されて実行が止まりました。何ともはやくだらないプログラムですね？──おっと，ここでバカにしてはいけません。我々が〝マシン語入門〟第一巻を読んでいる頃は，その方法すらわからなかったのです。これがくだらなく見えたのは，それだけ我々が進歩したということですね？　そう，我々は
　　　マシン語の中級者
なのです。

♥を表示するプログラムでしたら，〝マシン語入門〟第一巻でも散々やりましたね？　ところで，もういちど第８図を御覧になってください。同じ
　　　♥を表示する
にしても，その方法が異なっていることに気付かれるでしょう。

１文字出力ルーチン

第８図を見ますと，プログラムの中程に
　　　ＣＡＬＬ　２５７Ｈ
というのが見られます。これは，
　　２５７番地からのサブルーチンをＣＡＬＬ
する命令でしたね。それでは，そこにはどのようなサブルーチンが書かれているのでしょう？
「ＰＣ－8001　マシン語入門」第一巻で見ましたように，
　　　２５７番地付近は**ＲＯＭ**
になっていて，そこに
　　　BASICインタプリタ
が書き込まれています。したがって
　　　ＣＡＬＬ　２５７Ｈ
は，BASICインタプリタで使用しているサブルーチンをＣＡＬＬしていることになります。普通，このサブルーチンを
　　　システム・サブルーチン
と呼んでいます。
２５７番地からのサブルーチンは，非常に便利で有益なサブルーチンとなっていますから，その使い方を良く理解し，おおいに活用すると良いでしょう。

第11図　Ｇコマンドでプログラムの実行

〈１文字出力サブルーチン〉
　番　地：２５７Ｈ
　機　能：Ａレジスタに格納されているキャラクタ・コードをＣＲＴに出力する。

その使い方は？

それでは，
　　　１文字出力ルーチン
の使い方を，第８図のプログラムを見ながら確認してみましょう。
最初の
　　　ＯＲＧ　０Ｃ１００Ｈ
で，「スタート番地＝アセンブルを開始する番地」をＣ１００Ｈにセットしています。
（注）本書では，16Ｋシステム，32Ｋシステムどちらの方でも利用できますように，
　　　プログラムのスタート番地＝Ｃ１００Ｈ
　に統一して話しを進めていくつもりです。

続いて１行目

LD　A，0E9H
で，Aレジスタにキャラクタコード＝E9Hをセットしています。E9Hは♥**のキャラクタ・コード**でしたね（〝PC-8001　マシン語入門〟第一巻P.180，〝PC-8001キャラクタ・コード表〟を御覧ください）？
　これで♥を表示する準備が整いました。
　　CALL　257H
を実行すると，みごと♥が表示されてメインルーチン↗のC105番地に戻ってきます。最後に**定石の**
　　JP　5C66H
で**モニタにジャンプ**し，
　　　＊
が表示されてコマンド待ちとなります。
　以上までの手順，〝PC-8001　マシン語入門〟第一巻をマスターされた方なら，お茶の子サイサイですね？

トランプにも強さがある

　マシン語の中級者であるあなたに，なぜ第8図のような
　　簡単なプログラム
をお見せしたのでしょう？　もちろん，それには理由があります。
　もう一つ，次の例を考えてみましょう。
　今，われわれは
　　CALL　257H
というのを覚えました。それを使った例題です。

> システム・サブルーチン257Hを利用して，
> 　♠♥♦♣
> を表示するプログラムを作りなさい。

「アホクサ！」
　マア，マア，そういわないでもう少しお付き合いください。〝インベーダー〟マスターのために。
　ところで，
　トランプのマークには強さがある
というの御存知ですか？　それは，この問題のとおり，
　　♠♥♦♣
の順番です。PC-8001の「キャラクタ・コード表」を見ても，
　　♠──→E8H
　　♥──→E9H
　　♦──→EAH
　　♣──→EBH
の順に並んでいます。カシコイ（？）ですね？　なにせ，パーソナル・コンピュータですから。

　これでキャラクタ・コードが決まりましたから，これらのコードをAレジスタにセットして，
　　LD　A，0E8H
そして，サブルーチンをCALLする。
　　CALL　257H
これでまず♠のいっちょあがり。他の♥，♦，♣についてもまったく同様です。
　　LD　A，0E9H
　　CALL　257H
　　　∫
　最後にプログラムを止めてやるため，モニタにジャンプさせてやる。
　　JP　5C66H
これで完成です。

面倒なサブルーチン表

　でき上がったプログラムを，アセンブラ・リストで示したのが，**第12図**です。いかがですか？　手書きのリストより見やすいでしょう？
　続いて

```
                ;=================
                ;  PRINT ♠♥♦♣
                ;    (CALL 257H)
                ;=================
                ;
                        ORG  0C100H
                ;
C100 3EE8               LD   A,0E8H        ;LET A='♠'
C102 CD5702             CALL 257H
C105 3EE9               LD   A,0E9H        ;LET A='♥'
C107 CD5702             CALL 257H
C10A 3EEA               LD   A,0EAH        ;LET A='♦'
C10C CD5702             CALL 257H
C10F 3EEB               LD   A,0EBH        ;LET A='♣'
C111 CD5702             CALL 257H
C114 C3665C             JP   5C66H         ;GOTO MONITOR
                ;
                        END
```

第12図　4種類のマークを表示する

```
¥DC100,C116
C100 3E E8 CD 57 02 3E E9 CD 57 02 3E EA CD 57 02 3E
C110 EB CD 57 02 C3 66 5C
¥
```

第13図　Dコマンドで確認

```
¥GC100
♠♥♦♣
¥
```

第14図　Gコマンドで実行

DC100, C116↘

で入力の確認です（**第13図**）。そして

GC100↘

で実行させましょう。画面に

♠♥♦♣

が表示され，コマンド待ちとなりました（**第14図**）。予定どおりですね。

以上のように

CALL　257H

というのは非常に有用な命令です。そして，一つのまとまったプログラムを書こうとすると，プログラムのあちこちでこのサブルーチンを使うことになるでしょう。

ところで，257Hという番地，**覚えやすい**と思いますか？　もし，あなたがカシコイ人（失礼）なら，ヒョヒョイのヒョイと覚えてしまうでしょう。でも私は，おツムの関係で，いちいちこんな番地は覚えていられません。

そこで実際は，どこかの紙に

システム・サブルーチン表をメモ

しておき，プログラムを組みながら必要が生じるたびにそのメモをいちいちひっくり返す，ということになります。——これ，非常に面倒くさいのですよね。これも頭の悪い者の宿命でしょう。

ところで，アセンブラを使うと，この問題はワリと簡単に解決してしまいます。

番地に名前をつける

そこで登場するのが，

EQU命令

です。

まずあなたは，

257番地に**名前をつける**

ことから始めます。HANAKOでも，TONMAでも，DEKOBOKOでも，MOMOEでも，何でもかまいません。しかし，どうせならその**サブルーチンのイメージがわくように**，PRINTとか，CRTとか，適当なものが良いでしょう。ここでは，

OCRT

と名付けることにします。OCRだと，“光学式文字読取装置”（optical character reader）みたいな感じがしますが，

OUT　CRT（CRTに出力）

のつもりです。

名前が決まったら，

OCRT：EQU　257H

と書きます。つまり，

［名　前］：EQU［番　地］

という書式です。ついでに，この名前の意味を忘れないように，先に覚えた

；（コメント）

を使って注釈を書いておきましょう（**第15図**）。

どんどん使おうEQU

以上のように，

番地に名前をつけておく

と，もうサブルーチンの番地は忘れてしまって結構です。あなたの代わりにアセンブラが，記憶しておいてくれます。

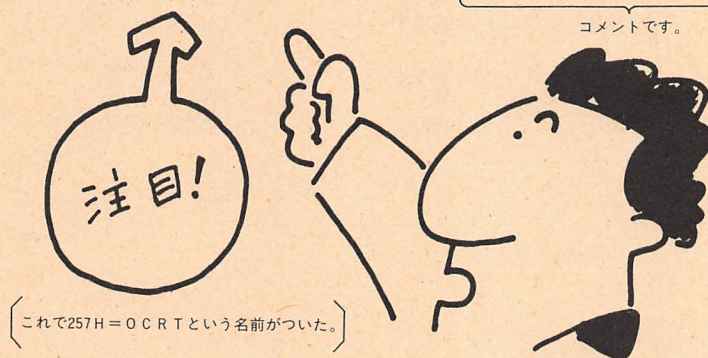
たとえばプログラムの途中で，

1文字出力ルーチン

を使いたくなったとします。すると今までだったら，「エーと，“1文字出力ルーチン”の番地は，どこだっけ？」

と，メモをあっちこっちひっくり返していました。そしてようやく所用の番地を見つけ出し，

CALL　257H　————①

とやるところでした。しかし，今やその必要はありません。

第15図　番地に名前をつける

番地の名前

だけを思い出せば良いのです。

OCRT

でしたね？　これならすぐ思い出せます。そしておもむろに

CALL　OCRT　――――②

と書けば良いのです。番地なんて必要ありません。

名前をＥＱＵで定義

しておきさえすれば，アセンブラは

①も②も同じ

に扱ってくれます。

それでは，先に見ました第８図のプログラムを，

ＥＱＵと名前

を使って書き換えてみましょう。**第16図**のリストがそれです。

Ｃ１０２番地：CALL　OCRT

になっていますね？　この第16図と第８図のリストの

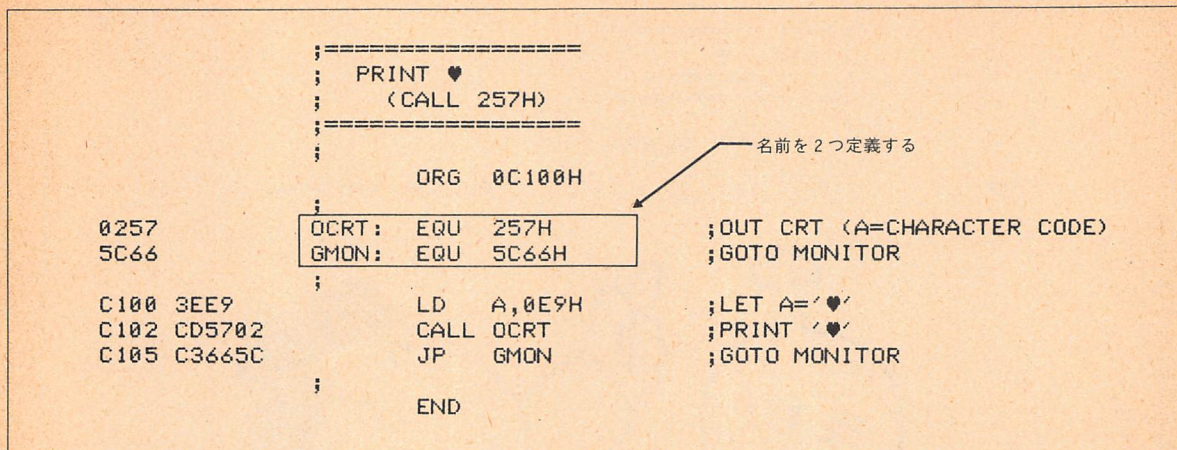
マシン語の部分を比較

してみてください。**まったく同じ**であることがおわかりになるでしょう。

さらにもう一つ。

第16図のプログラムには，もう一ヶ所

第16図　名前を使う

```
                    ;=================
                    ;  PRINT ♥
                    ;    (CALL 257H)
                    ;=================
                    ;
                          ORG   0C100H
                    ;
0257                OCRT:  EQU   257H          ;OUT CRT (A=CHARACTER CODE)
5C66                GMON:  EQU   5C66H         ;GOTO MONITOR
                    ;
C100 3EE9                 LD    A,0E9H         ;LET A='♥'
C102 CD5702               CALL  OCRT           ;PRINT '♥'
C105 C3665C               JP    GMON           ;GOTO MONITOR
                    ;
                          END
```


第17図　ついでにもう一つ

　　　番　地

が出ています。

　　　ＪＰ　５Ｃ６６Ｈ

です。この５Ｃ６６Ｈにも，ついでに名前をつけてやりましょう。この番地は，

　　　モニタのホット・スタート番地

ですから

　　　GMON

とでもつけましょう。

　　　GOTO　MONITOR

のつもりです。したがって

　　　GMON：EQU　５Ｃ６６Ｈ

と定義することができます。これを使って第16図をさらに書き換えると，**第17図**のようになります。やはり

　　　マシン語の部分は同じ

であることがわかります。

いかがですか？　ＥＱＵの威力は？　ちなみにＥＱＵは，

　　　ＥＱＵ＝equate

の略です。さらに前に見ましたＯＲＧは，

　　　ＯＲＧ＝origin

の略です。念のため。

ラ　ベ　ル

さて，前節までで

　　　名　前　の使い方

についておわかりいただけたことと（勝手に）思います。ところで，アセンブラにおいては，普通，名前のことを

　　　ラ　ベ　ル

と呼んでいます。

> **〈ラベル〉** label
>
> ロケーションにつける記号で，その同じ記号を用いてその位置を指定するのに用いる。

一般的に，ラベルとして用いる名前は，次のような制限があります。

① **１字目は，英字であること**
　ただし，他の文字の使用を認めているアセンブラもあります。

② **２字目以降は，英数字**（Ａ～Ｚの他１～９）

③ **長さの制限**
　３～８字位が多いようです。詳しくは，あなたのお持ちのアセンブラのマニュアルを見てください。なお，ちなみにラベルの長さを３文字しか認めていないアセンブラとは，かの有名な

　　　ＣＡＰ-Ｘ（通産省）

です。

今後，我々は

　　　ラ　ベ　ル

をジャンジャン用いていくことになります。良くここでその名前を覚えておいてください。念のため，もう一つ

```
                          ;=================
                          ;  PRINT ♠♥♦♣
                          ;    (CALL 257H)
                          ;=================
                          ;
                                 ORG  0C100H
                          ;
0257                      OCRT:  EQU  257H          ;OUT CRT(A=CHARACTER CODE)
5C66                      GMON:  EQU  5C66H         ;GOTO MONITOR
                          ;
C100 3EE8                        LD   A,0E8H        ;LET A='♠'
C102 CD5702                      CALL OCRT
C105 3EE9                        LD   A,0E9H        ;LET A='♥'
C107 CD5702                      CALL OCRT
C10A 3EEA                        LD   A,0EAH        ;LET A='♦'
C10C CD5702                      CALL OCRT
C10F 3EEB                        LD   A,0EBH        ;LET A='♣'
C111 CD5702                      CALL OCRT
C114 C3665C                      JP   GMON          ;GOTO MONITOR
                          ;
                                 END
```

第18図　ラベルを用いて書き換える

ＥＱＵを用いた例を掲げておきましょう。しつこく，しつこく。第12図のプログラムを，ラベルを用いて書き換えてみましょう。

ハイ，どうということはありません。**第18図**のとおりです。マシン語の部分は，まったく同じです。でも第12図に比べ，**ずっと見やすいプログラム**だと思いませんか？

もう一つの方法

もう**ラベル**という概念は，つかめたことと思います。ラベルを定義するには，ＥＱＵを用いて

〈ラベル〉：ＥＱＵ〈番　地〉

とするのでしたね。

ところで，ラベルを定義するには，**もう一つの方法**があります。それは，**ＯＲＧをまったく使いません**。この方法も良く出てきますので，ここでマスターしておきましょう。

今，仮に

ＬＤ　Ａ，０Ｅ９Ｈ

という命令があったとします。そしてこの命令の書かれている番地に，ラベルをつけたいとします。

今までの我々でしたら，**ＥＱＵを用いて**（仮にその番地をＣ１００Ｈ，またつけたいラベルが，MAINという名前だとします），

MAIN：ＥＱＵ　０Ｃ１００Ｈ
　　　ＬＤ　　Ａ，０Ｅ９Ｈ

または，

　　　ＬＤ　　Ａ，０Ｅ９Ｈ
MAIN：ＯＲＧ　０Ｃ１００Ｈ

とするでしょう。

(注) ＥＱＵによるラベルの定義は，**プログラム中，どこにあっても構いません**。ただまとめておくと便利ですから，前の方に置くことが多いだけです。

ＥＱＵを使わなくても

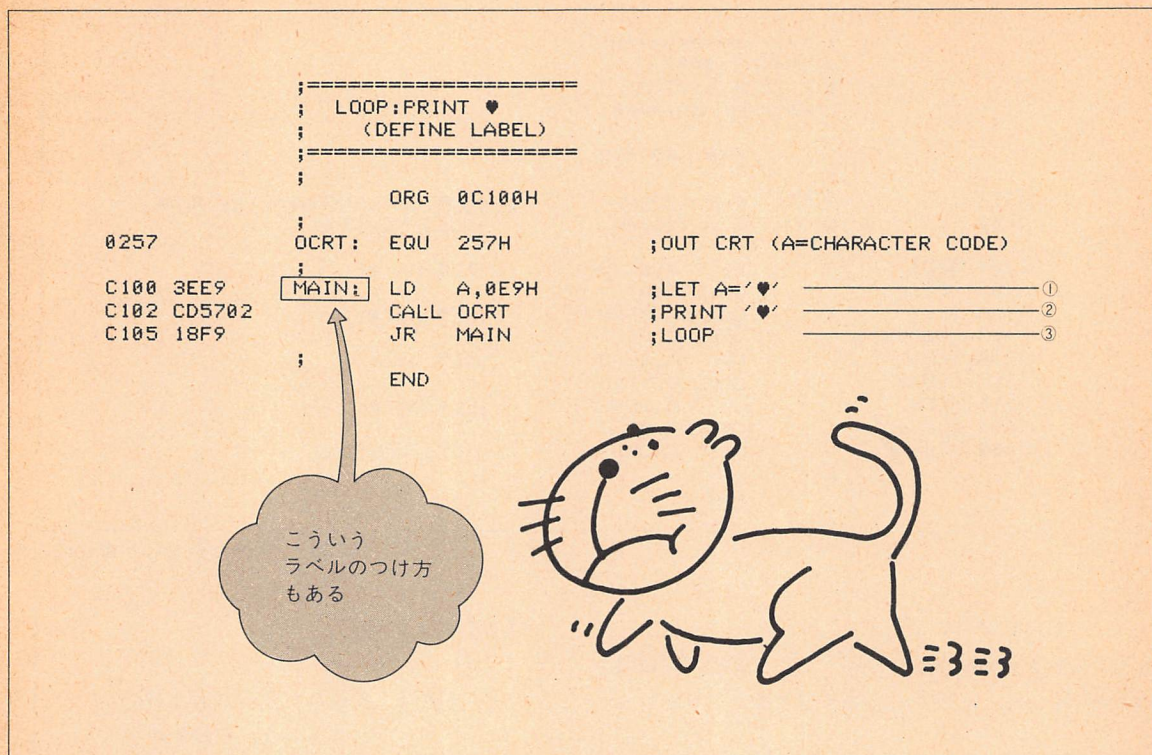
今の例でしたら，なにもわざわざＥＱＵを用いる必要はありません。ズバリ命令の前に

MAIN：ＬＤ　Ａ，０Ｅ９Ｈ

のように：（コロン）で**区切ってラベルを書けば良い**のです。

それでは，一つ例をお目にかけましょう。**第19図**のＣ１００Ｈを御覧ください（①の行です)。

ラベル**MAIN**

が**ＯＲＧ無し**で定義されています。これで

第19図　ラベルのつけ方，その2

MAIN＝Ｃ１００番地

としてアセンブラに記憶されます。したがってその2行下，Ｃ１０５Ｈの

ＪＲ　MAIN

は，

ＪＲ　０Ｃ１００Ｈ

と同じに解釈されるわけです。

ついでにこのプログラムの意味を考えておきましょう。まず①で

Ａレジスタ＝Ｅ９Ｈ

＝♥のキャラクタ・コード

がセットされます。そして②の

CALL　OCRT

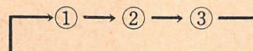
で，〝１文字出力ルーチン〟が呼び出され

♥がPRINT

されます。さらに③の

ＪＲ　MAIN

が実行されると，また最初に戻されてしまいます。つまりこのプログラムは，

┌→①→②→③┐
└────────┘

と繰り返される**恐怖の無限ループ**となっているのです。その間②によってＣＲＴ（モニタ・テレビのことですよ）に♥が出力され続けます。おおむねどういうプログラムか想像がつきましたか？

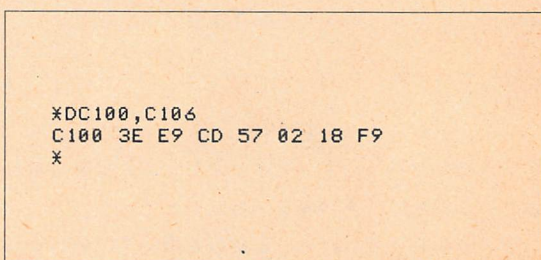
それでは実際にプログラムを入力し，走らせてみましょう。Ｄコマンドで確認し（**第20図**），

ＧＣ１００↘

でスタートです（**第21図**）。予想どおり画面上に♥が，ダァーッと出力されます。それは，やがて画面いっぱいに広がり，放っておけば永遠に続きます。これを止めるには，もはや

リセット・スイッチを押す

しかありません。

```
*DC100,C106
C100 3E E9 CD 57 02 18 F9
*
```

第20図　Ｄコマンドで確認

```
¥GC100
♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥♥
```

←これは，Ｇコマンド実行直後の様子です。やがてこれが，画面いっぱいに広がります。

第21図　無限ループの開始

ラベルのまとめ

以上，我々は

　　ラベル定義の二つの方法

をマスターしました。まとめますと，

> **〈ラベルの定義の仕方〉**
>
> ①　**ＥＱＵを用いる**
>
> 　　〈ラベル〉：ＥＱＵ　〈番　地〉
>
> ②　**ＥＱＵを用いない**
>
> 　　〈ラベル〉：〈命　令〉

たぶんお気付きだと思いますが，**ラベル定義のポイント**は，

　　：**（コロン）**

にあります。すなわちアセンブラは，

　　：の手前をラベル

と判断しているわけです。

最後に，**①と②の使い分け**について。

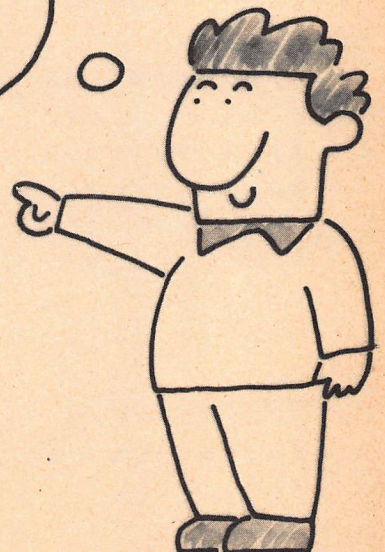
①に比べ②の方が楽ですから，通常は②の方を用います。しかし，

　　ラベルを定義したい番地が，自分の
　　プログラム内に無い

とき，たとえば

　　システム・サブルーチンの先頭番地

等にラベルをつけたい時は，②の使い方ができません。したがって，必然的に①のタイプで行くことになりまます。

以上が，①，②の使い分けです。ＯＫ？

〈リビング・ルーム〉

—ラベルの話題—

マシン語は，BASICに比べ，

プログラミングの能率

＝生産性

といった観点からみると，非常に見劣りがします。マシン語の**緻密性**ゆえのどさんくささ。かてて加えて，BASICのあの強力な

エディタ

が使えません。これは何を意味するかといえば，プログラムの

挿入・修正ができない

ということです。そこでハンド・アセンブルでは，あらかじめ**将来の修正を予測し**，

プログラム単位毎に

NOP（マシン語＝００H）

を入れたり，パッチを当てて修正したりします。あなたが雑誌等のマシン語のプログラムを見ていて

ＲＥＴ（マシン語＝Ｃ９H）

のあとに，００Hがズラッと並んでいたりするのはそのためです。

ところでマシン語でも，ひとたび

エディタ付きのアセンブラ

を使えば，話しは違ってきます。

テキスト・エディタのおかげで，マシン語レベルでも

プログラムの挿入・修正

が簡単にできるようになります。

(注) **エディタ**と**アセンブラ**とは，本来，まったく別の概念のものです。しかし現在発表されているアセンブラは，ほとんど

テキスト・エディタ付き

のものが多いため，

アセンブラ＝エディタ

または

アセンブラ≧エディタ

のように考えている人がいるようです。アセンブラとは，そもそも

ＬＤ　Ａ，３１Ｈ

のようなアセンブリ言語を

３Ｅ　３１

のように変換する機能を持つだけです。

たとえば，ＰＣ-8001の上位機種にあたるＰＣ-8800では，そのモニタ内部に

アセンブラ

を持っています。しかし，テキスト・エディタはついていませんから，本来の意味での

アセンブラそのもの

といえるでしょう。

さて，アセンブラにテキスト・エディタが加わった時，ある面では

BASIC以上

の機能を発揮します。それは，我々が今まで見てきたようにアセンブラには，

ラ　ベ　ル

という概念を持っているからです。

たとえば，BASICでは，

GOSUB　１０００

ということはできますが，

GOSUB　KEISAN

ということはできません。しかし，アセンブラなら，

CALL　KEISAN

でも

CALL　××××H

でもどちらでも可能なわけです。

一度，ラベルの味を覚えたら，数字の羅列である

番地とか行番号

とかは使う気になれません。あなたも，せっかくですからドシドシ

ラ　ベ　ル

を用いてプログラミングしましょう。

(注) 国産BASICの中には，従来よりラベルの使えるものが出ていました。しかし遅まきながらＰＣ-8001の上位機種であるＰＣ-8800でもラベルが使えるようになりました。今後発表されるBASICはラベルを使えるものが増えてくると思われます。おおいに期待したいところですね。

第3章

文字列出力ルーチンをめぐって

```
D300: D5 E5 CD 72 D5 06 3C CD 73 D9 E1 E5 CD 6B D5 21 ; 0A1D
D310: B9 E3 7E 2F 77 E1 D1 24 CD 8C D2 18 DF 01 00 01 ; 07BA
D320: 21 01 4B 78 FE 06 C8 E5 CD 49 D3 C5 06 32 CD 73 ; 07BC
D330: D9 21 B9 E3 7E 2F 77 C1 E1 E5 CD 62 D3 E1 25 78 ; 09C1
D340: 81 47 0E 00 CD 8C D2 18 DA C5 C5 E5 11 DD DF CD ; 08FC
D350: 72 D5 E1 C1 7C C6 05 67 10 F0 C1 3A C3 F3 A7 C0 ; 09AF
D360: 0C C9 C5 C5 E5 CD 6B D5 E1 C1 7C C6 05 67 10 F3 ; 09A4
D370: C1 C9 21 01 31 7C FE 0A C8 E5 06 05 CD 62 D3 E1 ; 07FC
D380: 25 E5 CD 90 D3 06 32 CD 73 D9 E1 CD 8C D2 18 E5 ; 0994
D390: C5 E5 11 DD DF CD 72 D5 E1 C1 7C C6 05 67 10 F0 ; 09DB
D3A0: 21 B9 E3 7E 2F 77 C9 21 23 98 22 CC F3 26 38 22 ; 06E7
D3B0: 44 F4 21 01 47 22 CD E3 CD 2C DB 21 CE E3 7E 3D ; 07D4
D3C0: 77 CD 1F DB 06 40 CD 73 D9 CD 8C D2 3A CE E3 FE ; 09B1
D3D0: 2E 20 E5 C9 3E 4C 32 AC E3 CD 12 D4 21 AC E3 35 ; 07DF
D3E0: CD F3 D3 06 0F CD 73 D9 CD 8C D2 3A AC E3 FE 05 ; 09B8
D3F0: 20 E7 C9 2E 14 CD 14 D9 3A AC E3 47 C5 11 02 E0 ; 0794
D400: 23 23 CD 1F D4 C1 60 2E 14 CD 0C D9 11 0C E0 C3 ; 06DB
D410: 1D D9 01 02 04 3A AC E3 67 2E 14 AF C3 4E D9 C5 ; 06CD
D420: E5 CD 2B D4 E1 01 78 00 09 C1 13 70 23 1A 77 04 ; 0610
D430: 13 23 1A A7 C8 70 23 77 04 13 23 18 F5 C5 CD 12 ; 05B4
D440: D4 21 AC E3 34 CD F3 D3 06 0F CD 73 D9 C1 CD 8C ; 0993
D450: D2 3A AC E3 B8 20 E6 C9 2A AD E3 C5 E5 CD 0C D9 ; 0A38
D460: 36 0F 06 0F CD 73 D9 36 00 E1 C1 2D CD 8C D2 7D ; 0720
D470: B8 20 E8 C9 25 25 2D CD 20 D7 AF 01 02 19 21 01 ; 05B1
```

でーびー・でーしー……

DATAの扱いにも注意を払って

　　　アセンブラ・リストの見方

について，もう少し話しが続きます。

　今度の話題も重要です。それは，アセンブラにおける

　　　DATAの扱い方

に関するものです。

　マシン語のプログラムは，大きく

　　　命令の部分

　　　DATAの部分

の二つに分けられます。このDATAを配置するのに

　　　DC：define constant

　　　DB：define byte

の二つの方法があることは，御存知ですか？　これについては，「PC-8001　マシン語入門」（第一巻）のP.110で軽く触れておきました。

　第一巻では主としてマシン語，それも

　　　ハンド・アセンブル

による方法を御紹介しました。ハンド・アセンブルの場合，

　　　アセンブリ言語──→マシン語

の変換は自分で行うわけですから，

　　　DATAの扱い

もいい加減で良かったわけです。しかし，アセンブラの場合，ここらあたりの**制約がきびしい**ですから，ここで，　アセンブラにおけるDATAの扱い方

を整理しておきましょう。

　まず，簡単な例からいきます。

　　　マ・イ・コ・ン

というカタカナをマシン語で表示することを考えてみましょう。

　いろいろな方法が考えられると思いますが，いずれにしても，〝**キャラクタ・コード表**〟から**キャラクタ・コード**を求める必要があります。

　　　マ──→CFH

　　　イ──→B2H

　　　コ──→BAH

ン──→DDH

が求めるコードです。これはマシン語から見れば

プログラム…………×

ではなく

DATA………………○

です。アセンブラに

C3　66　5C

というマシン語を作ってもらいたい時は，

JP　5C66H

と書けば良いのですが，マイコンのキャラクタ・コードである

CF　B2　BA　DD

は，どのように書けばアセンブラがマシン語のDATAに変換してくれるのでしょうか？

DBとDC

ここで困ってしまうのは，**DATAを定義する命令**が

アセンブラによってマチマチ

であるということです。しかし，いつまでそれを嘆いていても始まりません。ここではアセンブラを使う際困らないよう，DATAを定義する命令の代表的な使い方をいくつか御紹介しておきましょう。

まず，**DB**。

これは第一巻でも触れましたように

define byte

の略で，

1バイト分のDATAを定義する

といった意味です。最もきびしいアセンブラでは，16進数しか認めず，

DB　××H，××H，……

のように使います。このDB命令を用いて

マ・イ・コ・ン

のDATAを定義したのが，**第22図**です。

ところで，いちいち何かを表示するとき　マ──→CFH

イ──→B2H

⋮

のように**キャラクタ・コードに変換しなければならない**というのは，面倒だと思いませんか？　こんな**単純作業**こそコンピュータにやらせたいですね。そんなささやかな願いを，大抵のアセンブラはかなえてくれます。

そんな時に使うのが，**DC**です。

DCを第一巻では

define constant

と申し上げました。constantは，数学では**定数**という意味ですから，ここでは

define character

と考えた方が良いかもしれません。

DCの使い方は，

DC　'文字列'

または，

DC　"文字列"

というのが多いようです。

DCを用いて先程の

マ・イ・コ・ン

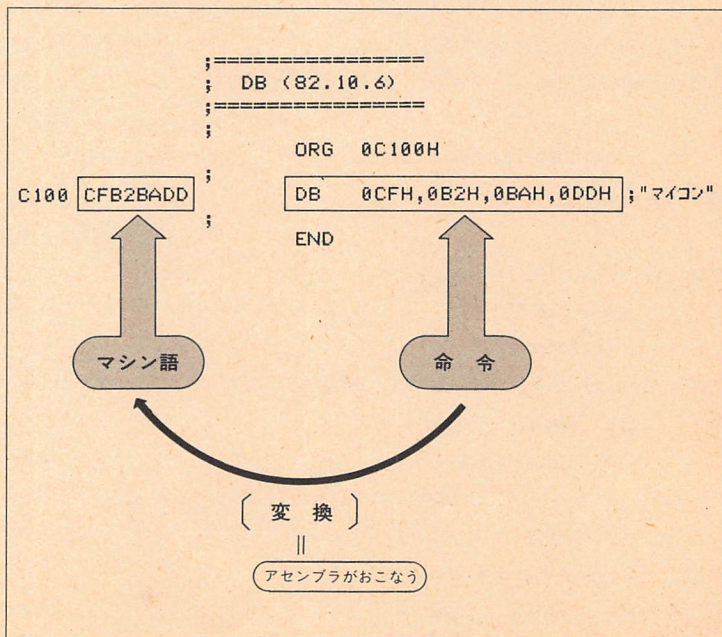
を定義してみましょう。**第23図**です。ここで注目していただきたいことは，あなたが命令を書く際には

文字をキャラクタ・コードに変換

する必要がない！

ということです。その変換は，

アセンブラ

があなたに代わってやってくれます。

第22図　DBでDATAを定義する

第23図　DCによる定義

文字列出力ルーチン

「PC-8001　マシン語入門」(第一巻) のP.118で，〝**文字列出力ルーチン**〞の機能を持つシステム・サブルーチンの先頭番地だけを紹介しておきました。せっかく前節で

　　　DBとDC

の使い方を覚えましたので，その練習を兼ねて，〝文字列出力ルーチン〞を使ってみましょう。

> **〈文字列出力ルーチン〉**
> **番　地**：52EDH
> **入　力**：HLレジスタに文字列の先頭番地をセットする。文字列の最後には，ENDマークとして00Hを置く。
> **機　能**：指定された文字列をテレビ画面に表示する。

もうあなたは，この説明だけでその使い方のすべてをのみこめるでしょう。そこで問題です。

> システム・サブルーチンの〝文字列出力ルーチン〞を使って
> 　　**　PC－8001　マイコン　**
> と表示するプログラムを作りなさい。

ラベルの乱用

まずは，

　　**　PC－8001　マイコン　**　という文字列を準備しましょう。もちろんキャラクタ・コードに変換する必要はありません。

DC命令を用いれば良いのです。

　DC　`**　PC－8001　マイコン　**'

で結構です。アセンブラ・リストでお見せしましょうか？　**第24図**です。

次にこのDATAの先頭番地を，HLレジスタにセットします。そのためには，DATA の先頭に

　　　ラ　ベ　ル

をつけておくと便利ですね。

　　　DATA

とでも名前をつけましょう。すると，

　　　DATA：DC　`**　PC－　〜　'

のようになります。アセンブラ・リストで見ますと，**第25図**のようになります。このDATA のマシン語の部分，

　　　2A　2A　20　50
　　　43　2D　38　30
　　　　　　〜

を

　　　文　字　列

といいます。念のため。

文字列の先頭番地にDATA というラベルをつけましたから，

　　　LD　HL,DATA

とやって，**HLレジスタに文字列の先頭番地をセット**してやります。これで〝文字列出力ルーチン〞をCALLする準備ができました。おっと，〝文字列出力ルーチン〞の先頭にも**ラベル**をつけておきましょう。

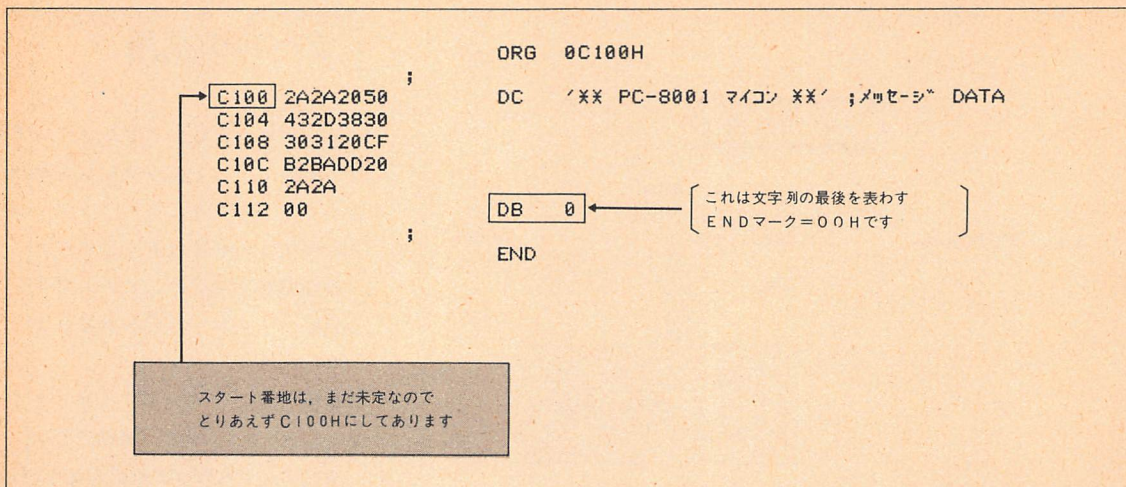
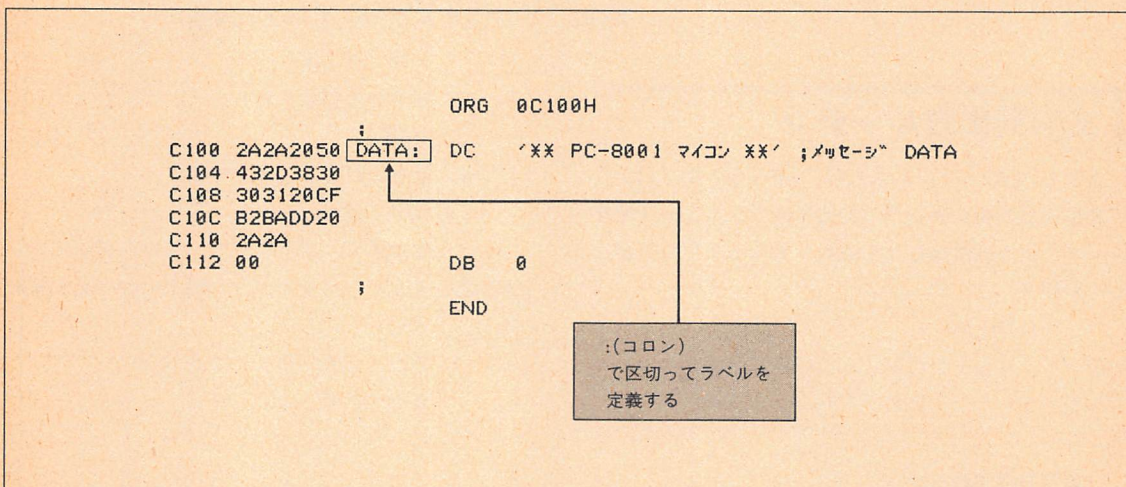
　　　MSG (message)

とでもつけることにします。

　　　MSG：EQU　52EDH

ですね。すると

　　　CALL　MSG

で〝文字列出力ルーチン〞をCALL できます。最後に**モニタのスタート番地**にも同じようにラベルをつけて

第24図　DATAの定義

第25図　ラベルをつける

おきましょう。

　　MON：EQU　5C66H

そして，

　　　JP　MON

でプログラム・ストップです。

二本建てでDATA定義

　こうしてできあがったプログラムが，**第26図**です。

　さっそくプログラムを入力して，あなたの目で御確認ください。

　　　SC100↘

　　　21

　　　09

　　　 ⌇

そしてDコマンドで確認です（**第27図**）。そして，いつものセオリーどおり，

　　　GC100↘

第28図を御覧ください。予定どおり

　　　＊＊　PC―8001　マイコン　＊＊

が表示されました。あなたのテレビ画面にもちゃんと表示されていますか？

　ここでアセンブラにおけるDATA 定義の仕方，その**バリエーションを一つ追加**しておきます。

　もう一度，第26図のアセンブラ・リストを御覧ください。C11B番地です。ここは，文字列の最後を示すため，

```
                 ;================
                 ;  CALL 52EDH
                 ;     (82.10.6)
                 ;================
                 ;
                       ORG   0C100H
                 ;
52ED             MSG:  EQU   52EDH         ;HE=POINTER,END MARK=0
5C66             MON:  EQU   5C66H         ;GOTO MONITOR
                 ;
C100 2109C1            LD    HL,DATA
C103 CDED52            CALL  MSG
C106 C3665C            JP    MON
                 ;
C109 2A2A2050    DATA: DC    '** PC-8001 ﾏｲｺﾝ **'  ;ﾒｯｾｰｼﾞ DATA
C10D 432D3830
C111 303120CF
C115 B2BADD20
C119 2A2A
C11B 00                DB    0
                 ;
                       END
```


第26図　プログラムの完成

```
*DC100,C11B
C100 21 09 C1 CD ED 52 C3 66 5C 2A 2A 20 50 43 2D 38
C110 30 30 31 20 CF B2 BA DD 20 2A 2A 00
*
```

第27図　Ｄコマンドで確認

```
*GC100
** PC-8001 ﾏｲｺﾝ **
*
```

第28図　プログラムの実行

００Ｈ

を**ＤＢ命令**で定義しているところです。これは，

ＤＣ　｀～′

で定義することができないため，別にＤＢを使っています。〝文字列出力ルーチン〟を使うとき，**必ず〝文字列〟を定義する**わけですが，ＤＢよりはＤＣの方が便利なため(キャラクタ・コードに変換する必要がないから)，

ＤＣ　｀文字列′

とやります。しかし，**文字列の最後には，必ず**

００Ｈ

を置かなければなりません。これは，ＤＣではできません。したがって，

ＤＢ　００Ｈ

とやります。すると，〝文字列出力ルーチン〟でＤＡＴＡを定義するには，

〝文字列〟──→ＤＣ

００Ｈ───→ＤＢ

のように**二本建て**にしなければならないことになります。これは面倒だと思いませんか？

一本のＤＢ命令で

制限のゆるいアセンブラでは，ＤＢ命令一つで

キャラクタ（文字列）

数（10進数，16進数）

の混在を許すものがあります。**コンマ**，で区切って，どのようにでも**DATA 定義**をできます。

第29図を御覧ください。

16進数……４１Ｈ，４２Ｈ，４３Ｈ，４４Ｈ

文字列……　｀ＡＢＣＤ′

10進数……６５，６６，６７，６８

を**一つのＤＢ命令で定義**しているのがおわかりになると思います。こういった

数　と　**｀文字列′**の混在

を許すアセンブラも存在するわけです。

なお第29図で

｀Ａ′＝４１Ｈ＝６５

であることはおわかりでしょうか？　展開されたマシン語をみてください。

第26図のプログラムのDATA定義部分を，**１本の**

```
                          ORG   0C100H
                 ;
C100 41422B44  DATA:      DB    41H,42H,43,44H,'ABCD',65,66,67,68  ;ABCD*4
C104 41424344
C108 41424344
                 ;
                          END
```

16進数　文字列　10進数

第29図　文字列と数を定義

```
                ;==================
                ;  CALL 52EDH-2
                ;       (82.10.6)
                ;==================
                ;
                        ORG   0C100H
                ;
52ED            MSG:    EQU   52EDH         ;HE=POINTER,END MARK=0
5C66            MON:    EQU   5C66H         ;GOTO MONITOR
                ;
C100 2109C1             LD    HL,DATA
C103 CDED52             CALL  MSG
C106 C3665C             JP    MON
                ;
C109 2A2A2050   DATA:   DB    '** PC-8001 ﾏｲｺﾝ **',0  ;ﾒｯｾｰｼﾞ DATA
C10D 432D3830
C111 303120CF
C115 B2BADD20
C119 2A2A00
                ;
                        END
```

文字列　10進数

第30図　DATAを1本のDB命令で定義する

DB命令で定義してみたのが，**第30図**のリストです。**マシン語の部分**が，第26図のものと同じになっていますね。

DATA定義のまとめ

長らくアセンブラにおける

DATAの定義

を見てきました。ここでまとめに入りましょう。

DATAには，大きく分けて

文　字　列

数

の二種類があります。前者を定義するのが，

DC……define character

であり，後者を定義するのが

DB……define byte

です。数は，さらに

10進数

16進数

の二種類に分かれます。両者を区別するため，

16進数にはHをつける

ことになっています。

DATAは，DC，DBで区別して定義していくのが普通ですが，中には**DBひとつで**

'文字列'も**数**も定義

できるアセンブラもありました。

(注)　アセンブラには，それこそ種々さまざまのものがあり，

DB→**DEFB**

DC→**DEFC**

と書くもの，また

DB→**DEFM** (memoryのこと)

DC→**DEFS** (stringのこと)

と書くのもあります。ここらあたりは，そのアセンブラのマニュアルで御確認ください。

第4章

中間言語を見る

```
D480: 0A CD 4E D9 3E B8 32 CB F3 32 43 F4 01 02 0F 11 ; 0670
D490: 81 E2 21 01 14 C3 9C DD 3A AC E3 A7 CB CD D1 D4 ; 097F
D4A0: 21 AC E3 35 18 0D 3A AC E3 FE 4C CB CD D1 D4 21 ; 087B
D4B0: AC E3 34 2E 17 CD 14 D9 3A AC E3 47 C5 11 02 E0 ; 078A
D4C0: CD 2F D9 C1 60 2E 17 CD 0C D9 11 0C E0 C3 1D D9 ; 07A3
D4D0: 00 01 02 04 3A AC E3 67 2E 17 AF C3 4E D9 AF 32 ; 05F6
D4E0: C4 E3 21 FA D4 E5 3A BA E3 A7 CA 07 D5 FE 01 CA ; 0A68
D4F0: F3 D5 FE 02 CA 52 D6 C3 9A D6 3A C4 E3 A7 C0 AF ; 0AE4
D500: 2A C2 E3 23 77 18 D7 2A C2 E3 7E FE FF CA 95 D5 ; 09D6
D510: A7 20 05 CD 32 D5 18 EF 06 0B C5 CD 4A D5 30 08 ; 06A1
D520: 25 ED 5B BF E3 CD 72 D5 21 BE E3 7E C6 05 77 C1 ; 0966
D530: 10 E8 21 BD E3 35 35 3A BC E3 32 BE E3 21 C1 E3 ; 0894
D540: 35 7E CD 87 D5 21 C2 E3 35 C9 2A BD E3 CD 5D D5 ; 0969
D550: D0 E5 CD 6B D5 E1 3E FF 32 C4 E3 37 C9 E5 24 24 ; 09E6
D560: CD 0C D9 7E E1 FE EE 37 C8 3F C9 01 02 05 AF C3 ; 087E
D570: 4E D9 D5 CD 0C D9 D1 3A B9 E3 A7 CA 1D D9 EB 01 ; 09A8
D580: 0C 00 09 EB C3 1D D9 E6 01 C8 2A BF E3 01 E8 FF ; 081C
D590: 09 22 BF E3 C9 21 BC E3 35 21 C4 E3 36 FF CD AF ; 0904
D5A0: D5 CD D0 D5 26 00 CD D6 D5 C8 21 BA E3 34 C9 21 ; 0989
D5B0: B9 E3 7E 2F 77 C3 B8 D5 2A BB E3 22 BD E3 21 DD ; 0998
D5C0: DF 22 BF E3 3E 05 32 C1 E3 21 B8 E3 22 C2 E3 C9 ; 0908
D5D0: 01 03 06 C3 15 DA AF 32 C5 E3 3A BB E3 6F 06 05 ; 0697
D5E0: CD 5D D5 30 05 3E FF 32 C5 E3 2D 2D 10 F2 3A C5 ; 07A6
D5F0: E3 A7 C9 2A C2 E3 7E FE FF 28 42 A7 20 05 CD 32 ; 08D2
```

また・また　なつかしい　もじれつ……

奇妙な文字列

次の話題は，最初に実験から入っていくことに致します。

まずは，第31図のプログラム・リストのマシン語をウムもいわずに入力してください。

(注)　このリストは，32Kシステム用のものです。もしあなたのＰＣ－8001が，まだＲＡＭを増設して

```
                    ;================
                    ;  DW (82.10.6)
                    ;================
                    ;
                            ORG   0C100H
                    ;
52ED                MSG:    EQU   52EDH          ;HE=POINTER,END MARK=0
5C66                MON:    EQU   5C66H          ;GOTO MONITOR
                    ;
C100 2A0DC1                 LD    HL,(TEXT)      ;HL=TEXT TOP
C103 23                     INC   HL             ;CANCEL LINK POINTER
C104 23                     INC   HL
C105 23                     INC   HL             ;CANCEL LINE NUMBER
C106 23                     INC   HL
C107 CDED52                 CALL  MSG
C10A C3665C                 JP    MON
                    ;
C10D 2180           TEXT:   DW    8021H   <-- 2バイトのＤＡＴＡ定義   ;N-BASIC TEXT TOP
                    ;
                            END
```

第31図　DW実験プログラム

```
¥SC100
C100 F1-2A 12-0D E5-
```

第32図　Sコマンドでプログラムを入力していく

```
¥DC100,C10E
C100 2A 0D C1 23 23 23 23 CD ED 52 C3 66 5C 21 80
¥
```

第33図　Dコマンドで確認

いないようでしたら

C10D番地：80H──→C0H

に変更して入力してください。

Sコマンドで入力します(**第32図**)。そしてDコマンドで確認しましょう(**第33図**)。おっと，ここで**プログラムを実行するのは待ってください**。ちょっと準備が必要です。

まず，コントロールBで

BASICのコマンド・レベル

に戻して，

CLEAR　300，& HC0FF↘

と入力してください(**第34図**)。これはBASICにより，**入力したマシン語が壊されるのを防ぐ**ためです。

```
clear 300,&hc0ff
Ok
■
```

第34図　マシン語のプログラムを保護する

続いてBASICのプログラムを入力してください。

10　REM　PC-8001　マイコン

と入力します(**第35図**)。

これで準備ができました。プログラムを走らせましょう。

MON↘

で**マシン語のコマンド・レベル**に。そして

GC100↘

でプロクラム・スタートです。

「ややっ！　変なものが！」

表示されました。何だ，コリャ（**第36図**）？

2バイトのDATA定義には注意を

以上を体験していただいたところで説明に入ります。

先に，

アセンブラにおけるDATA定義

の方法を見ました。そこに展開されたのは，

1バイトのDATA

を中心としたものです。しかしマシン語では，

2バイトのDATA

を扱うことも多いですね。

たとえば

第35図　BASICのプログラムを入力

第36図　プログラムの実行！

番　地

等は，2バイトのDATAです。したがってマシン語でプログラムを組んでいくと

2バイト単位でDATAを定義

する必要が良く起こります。

今，仮に

1234番地

というDATAを定義したいとします。我々の既知の知識でこれを実現しようとすると，たぶん

DB　12H，34H

のようにやるでしょうね――ダメ！

ダメなのです，これでは。80系のCPUでは。

80系のCPUでは

「PC―8001　マシン語入門」(第一巻)で**ハンド・アセンブル**した際，

2バイトのDATAを扱うときは，

上位と下位を逆転させた

ことを思い出してください。

ところで2バイトのDATAを定義するには，実は，

上位・下位をどの順においても構わない

のですが，そのあとで，80系の種々の命令を実行するには，

下位――→上位

の順に置いておいた方が便利なのです。たとえば，前節の例でしたら

DB　34H，12H

の方が**ベター**なのです。

〈秘　伝〉

80系のCPUで2バイトのDATAを定義するには，

下位――→上位

の順に並べる方がベター。

さて実際問題として，アセンブラで

2バイトのDATA定義

をするには，便利な命令が用意されています。

DW命令

がそれです。DWというのは，

define word

の略です。**ワード（word）**というのは，

メモリにおけるデータの基本単位

で，1ワードが何ビット分に相当するかはCPUによって異なります。8ビットCPUでは，普通

1ワード＝2バイト

で換算しています。JISによるワードの定義は，次のとおりです。

〈ワード〉

（＝語＝word）

何らかの目的から一つのものとして考えると都合がよい文字列。

DW命令の威力

DW命令を使うと，DATAの

上位と下位に気を使う

必要がなくなります。たとえば先の例ですと，

DW　1234H

で結構です。アセンブラは，これを

34H，12H

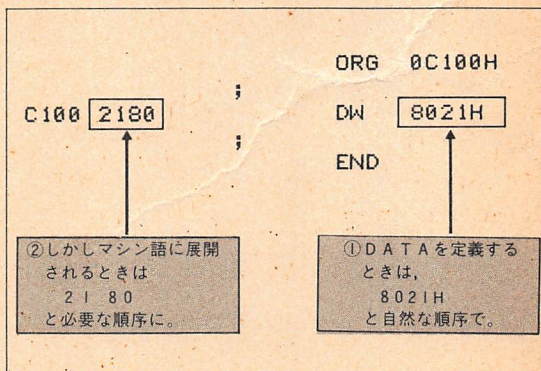
のように変換してくれます。

アセンブラ・リストによる例をお見せしましょう。**第37図**を御覧ください。アセンブリ言語でDATAを定義する際は，

8021H

のように**自然な順序**（8021番地です）で書いてかまいません。しかし，これをアセンブラを使ってひとたびマシン語に変換すると，

21H，80H

のように80系CPUにとって都合の良い順番に変換してくれます。

第37図　DWで2バイトのDATAを逆にする

N-BASICのメモリ格納状態

DW命令の威力を御理解いただいたところで，懸案事項である第31図のプログラムについて，見ていくことに致しましょう。

まずC10D番地を御覧ください。

DW　8021H

として，**DW命令**が使われています。このプログラムを理解するカギは，この

8021H

にあります。と申しますのは，実はこのプログラムは，

N－BASICのテキストがメモリに格納される様子

を表示させようとするものです。そして

8021H

は，そのことに深く係わりを持つからです。したがいまして第31図のプログラムを知るには，

N-BASICのメモリの状態

を知る必要があります。

これから述べますことは，

本文の流れとは関係ありません！

のでサラリと触れることにします。したがいまして，あなたもサラリと読み流してください。理解しなければならない理由は，サラサラありません。

BASIC 1行の構成

そこで**第38図**を御覧ください。

あなたがBASICでプログラムを入力していくと，そのプログラムは，

メモリに格納

されます。それでは、一体，プログラムはメモリのどこ（WHERE）？

に格納されるのでしょう。

――答。それが，

8021番地

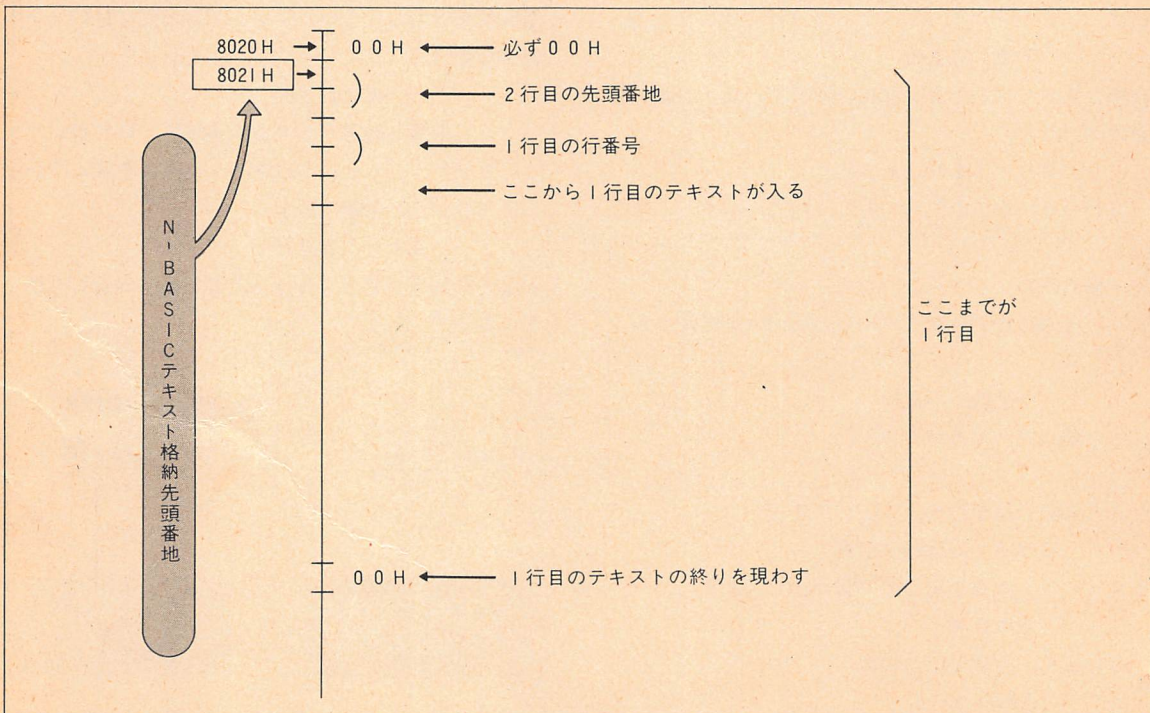
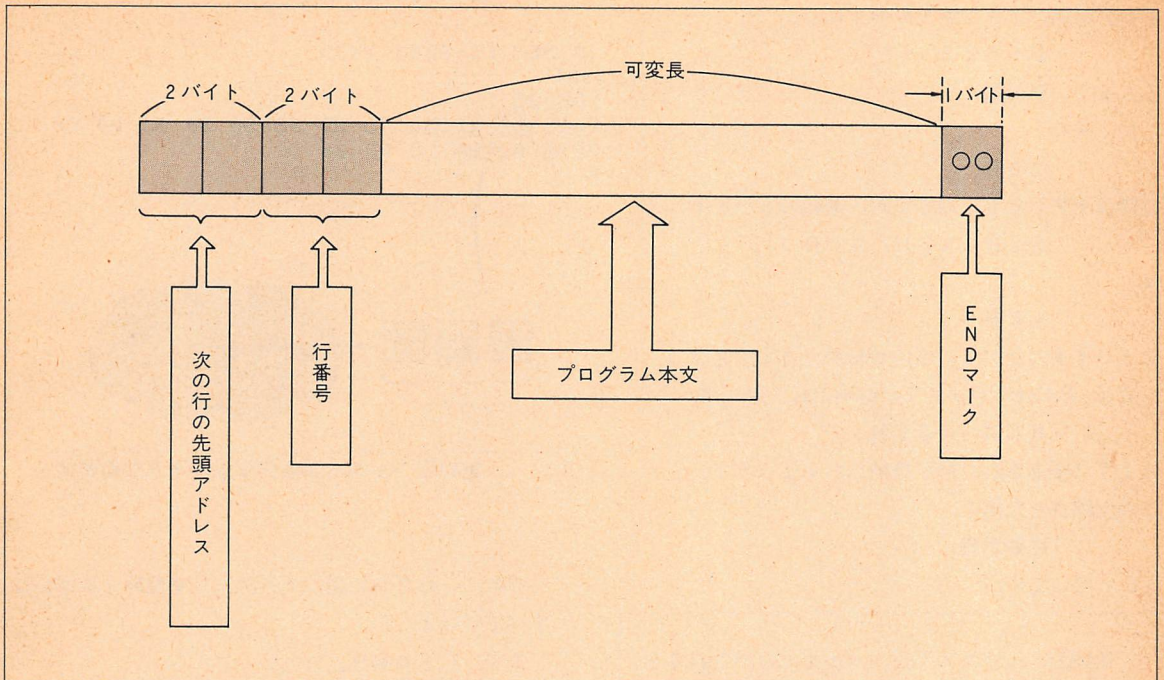
です。BASICでプログラムを入力していくと，そのプログラムは，

8021番地

から格納されていきます。

(注)　ただし，今のは**32Kシステム**の話であって，**16Kシステム**の場合，BASICのプログラムは，

C021番地

から格納されます。したがいまして，これから述べますことについても16Kシステムの方は，

第38図　N-BASICのテキストの格納状況

第39図　1行の構成

80⟶C0

のように置き換えてお読みになってください。

続いて**第39図**を御覧ください。BASICのプログラムがメモリに格納されるときの

1行の構成

が図式されています。

最初の2バイトには，次の行が格納されるアドレスが入っています。また**その次の2バイト**には，その行の行番号が格納されます。それは，

2バイトの16進数

に変換された形で入っています。たとえば，

行番号＝10

なら，

10＝A

ですから，これを2バイトに直し，

000AH

さらにこの上位と下位をひっくり返し，

0A00H

の形で格納されるわけです。

次の**5バイト目**から，テキストの本文が格納されます。特別な形で。

中間言語

たとえば，

REM PC-8001 マイコン

の場合，どのように格納されると思いますか？

容易に予測されるのが，

R―――――52H
E―――――45H
M―――――4DH
スペース―――20H
P―――――50H
C―――――43H
⟆

のように**キャラクタ・コード**に変換し，

52H，45H，4DH，20H，……

のように格納されるのではないか，と想像されることです。

本当かどうかは，実際に確かめてみればよいのです。BASICのコマンド・レベルで

10 REM PC―8001 マイコン

と入力し，

MON↘

でマシン語のコマンド・レベルにし，

D８０２５↘

とします（第40図）。

良く見ると，

２０H

以降は予想どおりなのですが，最初の**３バイト**

５２H，４５H，４DH

は

８FH

の**１バイト**に化けているのがわかります。どうやらN-BASICがメモリに格納される際には，

REM⟶８FH

のように圧縮された形で格納されるようです。

この圧縮された語を

中間言語

といっています。

第40図　BASICプログラムのメモリ格納状況

> BASIC のテキストが，メモリに格納されるとき，
>
> 中間言語
>
> という圧縮された形で格納される

LD HL，(TEXT)

そこでもう一度，第39図に戻ってください。

前節で見ましたように，テキストの本文は

特別な形

でメモリに格納されます。図の

プログラム本文

という所がそうです。この部分の長さは，文の長さやマルチ・ステートメントを使うことにより変わってきます。

そして最後に

その行の終りを示す

ために

００H

が入ります（ENDマーク）。

以上で

BASIC１行の構成

がわかったところでもう一度第38図を御覧ください。今度は先程見た時より，理解しやすいのではないかと思います。

さあ，これで第31図のプログラムを理解する準備はすべて整いました。

まず，C１００H。

LD　HL，(TEXT)

とあります。TEXTは，ラベルで下の方，C１０DHにありますね。

C１０DH＝２１H

C１０EH＝８０H

この２バイトのDATAが，

HL⟵８０２１H

のようにHLレジスタにセットされます。つまり，

HL＝N-BASICのテキスト先頭

になったわけです。

(注)　この部分がわかりにくい人は，**第41図**を御覧ください。

LD　HL，(……)

という命令は，

HLレジスタに２バイトの
DATAを代入する

命令です。その２バイトとは，(　)の中に示される番地に記憶されているところのDATAです。

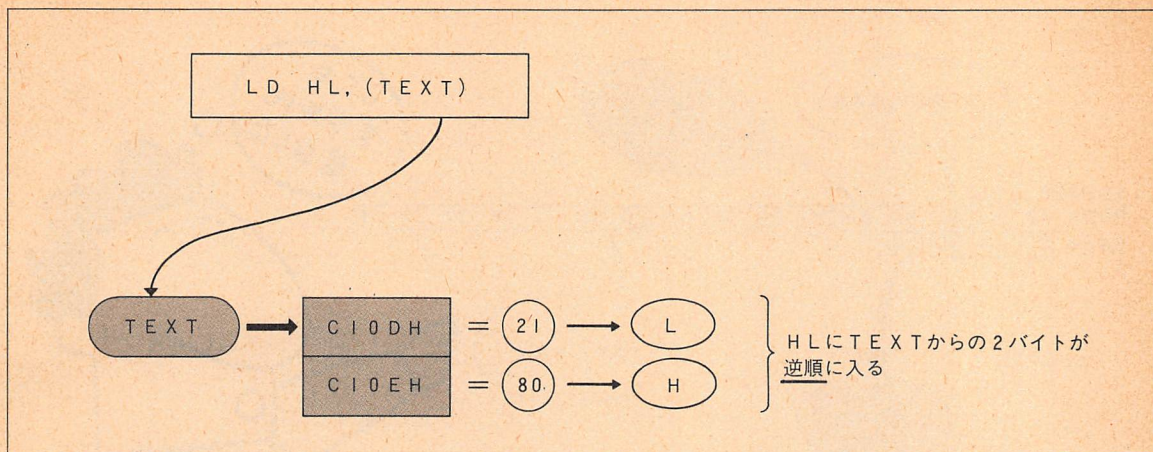
LD　HL，(TEXT)

でしたら，

TEXT＝C１０D

ですから，C１０D番地のDATAがHLレジスタに代入されます。ところが，

C１０D番地だけでは１バイト分

のDATAしかありません。そこで，その次のC１０

第41図　HLに2バイトのDATAを入れる

E番地のDATAも使われます。そして代入のされ方は，

のように

逆　順!

に代入されます。ここらあたりを，第41図を見ながら，良く納得してください。

中間言語＝文字列？

さて，

HLレジスタ←BASICの1行目の先頭

がセットされました。次に

INC　HL

が4つ並んでいます（第31図のプログラムですよ）。

INC　HL

は，**HLの値を一つ大きくする**命令でしたね。したがいまして

HL←HL＋4

が実行されたことになります。最初の

＋2

で〝次の行のアドレス〞を示す2バイトがスキップされます。そして，次の

＋2

で〝行番号〞がスキップされます。結局

HL＝8025H

を指すことになりました。この番地は，

BASIC本文の先頭番地

でしたね？　HLが，ここを指しながら次のC107番地で

CALL　MSG

をやっています。ということは，

〝文字列出力ルーチン〞

をCALLしているわけです。すると，どういうことが起こるでしょう？

〝文字列出力ルーチン〞は，HLの指しているところを文字列と見なし，そこからのDATAを

ENDマーク＝00H

が現われるまでテレビ画面に出力されるというものでした。とすると，ここでは何と

中間言語に圧縮されたBASICの
テキスト

を文字列と見なし，テレビ画面に出力してしまうのです（**第42図**）。

これは，たまたま

文字列の最後──→00H
テキストの最後──→00H

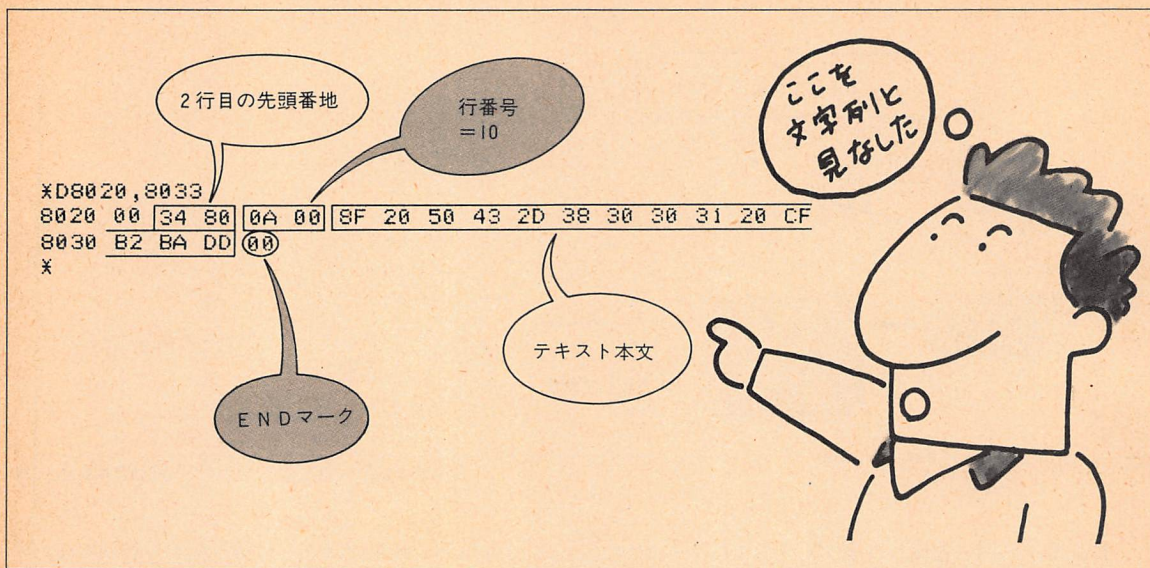
のように一致していたから，うまく表示されたのです。

中間言語を見る

そこでもう一度第36図の実行結果を御覧になってください。

＋　PC-8001　マイコン

と表示されています。今やあなたは，この奇妙な

＋

第42図　第31図のプログラムを追って

の正体がお分かりになると思います。

　　　　＋＝８ＦＨ（キャラクタ・コード）

です。

　　　　８Ｆ＝ＲＥＭの**中間言語**

でしたね？　つまり，本来は文字ではない中間言語を，〝文字列出力ルーチン〟が**むりやり〝文字〟と解釈**し，テレビ画面に表示してしまったものだったのです。

　これであなたは，第31図のプログラムが，何を狙ったものかもうおわかりでしょう？　このプログラムは，
BASICがプログラムをメモリに格納する際，
　　　　どのような形で中間言語化させる？
かを，**目で見える形**で表示させようとしたものです。あなたも，いろいろなBASICのプログラムを入力し（もちろん1行目だけ！），このプログラムを走らせてみてください。だんだん，どのように

　　　　BASICのプログラムが中間言語化
　　　　される

かがわかってくるでしょう。興味のある方は，ぜひおためしください。

再びなつかしい文字列に

　アセンブラ・リストの見方，だんだん終盤が近づいてきました。ここで次の話題に移る前に，〝ＰＣ―8001　マシン語入門」〟（第一巻）P.118で提起しておいた課題にお答えしておきましょう。

　それは，１８３８Ｈから入っている

　　　　N-BASICスタート時のメッセージ

を，〝文字列出力ルーチン〟を使って表示させようとするものでした。ところが，我々が第一巻で作った〝文字列出力ルーチン〟では，

　　特殊記号
　　　　CR………０ＤＨ
　　　　LF………０ＡＨ

の関係で一行に表示されてしまいました。そして第一巻では，そのことについて

「したがって現在の文字列を，**ＲＯＭ内の文字列表示ルーチン**を使えばこの文字列が正しく表示されるのです。ちなみにＲＯＭ内のそのルーチンは，５２ＥＤ番地から始まっています。（中略）５２ＥＤ番地に変えて実験してみてください。なつかしい文字列が，正しく表示されることでしょう。」

ということで，あなたに預けた形になっています。

　ＲＯＭ内のシステム・サブルーチン

　　　　〝文字列出力ルーチン〟

については，もうお馴染みですね？　そこで，軽くこの課題に挑戦しておきましょう。

　プログラムは，**第43図**のとおりです。チョチョイのチョイ，とできあがりです。ラベルを多用していることに御注目ください。

```
                    ;==================
                    ;  CALL 52EDH-3
                    ;      (82.10.14)
                    ;==================
                    ;
                          ORG   0C100H
                    ;
1838                DMSG:  EQU   1838H        ;DATA "N-BASIC"
52ED                MSG:   EQU   52EDH        ;HE=POINTER,END MARK=0
5C66                MON:   EQU   5C66H        ;GOTO MONITOR
                    ;
C100 213818               LD    HL,DMSG
C103 CDED52               CALL  MSG
C106 C3665C               JP    MON
                    ;
                          END
```

第43図　システム・サブルーチンを使って

そして，いつものようにマシン語を入力し，

DC100，C108↘

で確認です（**第44図**）。そして

GC100↘

でプログラム発射！（**第45図**）

「やった！」

なつかしい文字列が**二列**に表示されました。さらにしつこく，4回も実験してみました（**第46図**）。何回やっても，小気味良く成功します。やはり，我々は

マシン語の中級者（？）

になったのですね。

```
*DC100,C108
C100 21 38 18 CD ED 52 C3 66 5C
*
```

第44図　Dコマンドで確認

```
*GC100
NEC PC-8001 BASIC Ver 1.2
Copyright 1979 (C) by Microsoft

*
```

← 2列に正しく表示された！

第45図　なつかしい文字列が正しく二列に表示

```
*GC100
NEC PC-8001 BASIC Ver 1.2
Copyright 1979 (C) by Microsoft

*GC100
NEC PC-8001 BASIC Ver 1.2
Copyright 1979 (C) by Microsoft

*GC100
NEC PC-8001 BASIC Ver 1.2
Copyright 1979 (C) by Microsoft

*GC100
NEC PC-8001 BASIC Ver 1.2
Copyright 1979 (C) by Microsoft

*
```

第46図　4回繰り返えす

第5章

ワーク・エリアの設定

```
D600: D5 18 F0 06 0B C5 CD 4A D5 30 08 2C ED 5B BF E3 ; 07ED
D610: CD 72 D5 21 BE E3 7E C6 05 77 C1 10 E8 CD 32 D5 ; 0923
D620: CD D0 D5 2A BD E3 2C 2C 2C 2C 3A C1 E3 A7 28 08 ; 07A1
D630: FE 03 D0 3E D8 C3 60 D9 3E 58 C3 60 D9 21 BB E3 ; 0934
D640: 34 3E 32 23 77 21 C4 E3 36 FF CD AF D5 21 BA E3 ; 084A
D650: 34 C9 2A C2 E3 7E FE FF 28 26 A7 20 05 CD 32 D5 ; 0835
D660: 18 F0 06 0B C5 CD 4A D5 30 08 24 ED 5B BF E3 CD ; 07DD
D670: 72 D5 21 BE E3 7E D6 05 77 77 C1 10 E7 C3 32 D5 ; 08D2
D680: 21 BC E3 34 21 C4 E3 36 FF CD AF D5 CD D0 D5 26 ; 09DA
D690: 4B CD D6 D5 C8 21 BA E3 34 C9 2A C2 E3 7E FE FF ; 0A90
D6A0: 28 2B A7 20 05 CD 32 D5 18 F0 06 0B C5 CD 4A D5 ; 06BD
D6B0: 30 08 2C ED 5B BF E3 CD 72 D5 21 BE E3 7E D6 05 ; 087D
D6C0: 77 C1 10 E8 CD 32 D5 CD D0 D5 C3 23 D6 21 BB E3 ; 09F1
D6D0: 34 3E 19 23 77 21 C4 E3 36 FF CD AF D5 AF 32 BA ; 080E
D6E0: E3 C9 CD 03 D7 E5 CD 20 D7 E1 E5 CD 6B D5 E1 CD ; 0B7D
D6F0: 4A D7 21 00 00 22 AD E3 21 B1 E3 35 C0 3E 02 32 ; 0610
D700: AA E3 C9 2A AD E3 E5 CD 0C D9 7E E6 0F E1 28 03 ; 0926
D710: 25 18 F3 E5 24 24 CD 0C D9 7E E1 FE EE C8 2D C9 ; 0918
D720: CD 0C D9 11 16 E0 E5 CD 1D D9 CD 3E D7 06 10 CD ; 0826
D730: 73 D9 E1 11 22 E0 CD 1D D9 06 20 C3 73 D9 CD 4E ; 0853
D740: 0D 06 0A CD 73 D9 AF C3 4B 0D 3A BB E3 D6 03 BD ; 076E
D750: 38 07 D6 04 BD 38 07 18 0A 01 0A 00 18 08 01 14 ; 0277
D760: 00 18 03 01 32 00 2A C6 E3 09 22 C6 E3 C3 7C D8 ; 060C
D770: 3E 37 32 B1 E3 21 B2 E3 34 01 01 05 21 B8 E3 71 ; 0659
```

りょういき　の　かくほ……

領域を確保する？

アセンブラ・リストにおける命令，残りは，あと二つです。その一つ目は，

DS(define storage)

命令です。

> **〈DS命令〉**
> **書　式**：DS　n
> （nは，バイト数）
> **目　的**：nバイトの領域を確保する

この命令は，最初，その意味がわかりにくいかもしれません。使い方は，たとえば

DS　4

のようにします。すると，

4バイトの領域

が確保されます。といっても，この

領域を確保する

という意味が，なかなかとらえにくいですね？

例をあげましょう。**第47図**を御覧ください。

C100H＝3EH
C101H＝77H
C102H＝47H
C103H＝76H

の4バイトから成る，何の変哲もない，クダラナイプログラムです。このプログラム自体には，特に何の意

```
                    ;====================
                    ;  DS-1 (82.10.15)
                    ;====================
                    ;
                            ORG   0C100H
                    ;
C100 3E77                   LD    A,77H
C102 47                     LD    B,A
C103 76                     HALT
                    ;
                            END
```

第47図　何の変哲もないプログラム

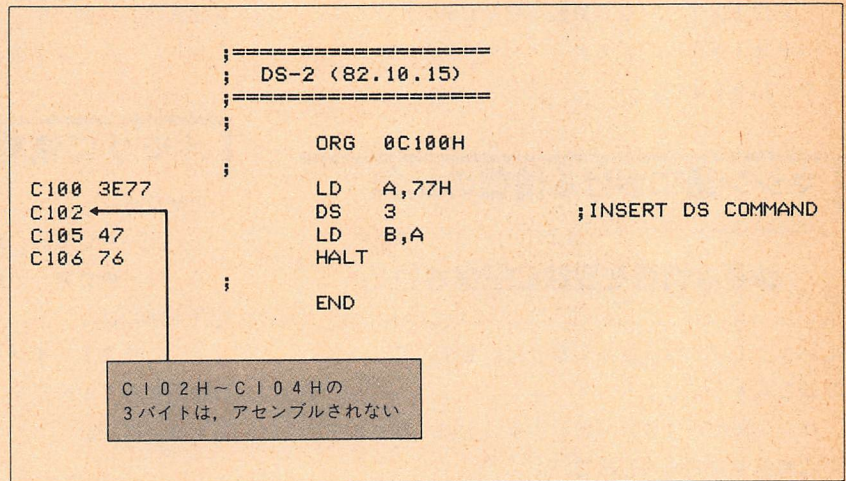
味もありません。こんなプログラムを走らせるような，バカなことはしないでくださいよ。

nバイトの空白

ところで，この
1行目：LD　A，77H
2行目：LD　B，A
の間に，
DS　3
という，**3バイトの領域を確保する命令**を入れてみます。その上でアセンブルすると，どういうことになるでしょう（**第48図**）。

第48図　ＤＳ命令を追加すると

第49図　ＤＳ命令挿入後のアセンブル・リスト

第49図が，アセンブルしたものです。良く番地とマシン語の関係を見てください。今度は，
C100H＝3EH
C101H＝77H
C102H＝
C103H＝
C104H＝　この3バイトは，マシン語に変換されない！
C105H＝47H
C106H＝76H
のように，
C102H～C104H
は，**マシン語に変換されない**，すなわち
3バイトの空白の部分
ができました。これが，
3バイトの領域を確保する
という意味の実態です。
まとめますと，
DS　n
を実行すると，
nバイトの領域が確保
され，マシン語の中に
nバイトの空白の部分
が生ずるのです。

誰がために"DS命令"は存在する

さあ，
ＤＳ命令＝領域の確保
の意味が，おぼろげながらわかったとして，こんな命令，一体
何に使う？
のでしょうね。やはり先を急ぐよりは，このことをハッキリさせておくことに致しましょう。

ＢＡＳＩＣでプログラムを作るとき，いろいろな**情報を記憶**させておく必要が起こりますが，そんな時，あなたはどうしていますか？　たとえば，GAMEのプログラムを作る時には，
得　点
等の情報を記憶させておかなければなりません。
普通，こんな時は**変数**を用意し，そこに記憶させて

おきます。たとえば

変数ＳＣＯＲＥ

に得点を記憶させるとして，

ＳＣＯＲＥ＝９８０

なら，

現在の得点＝９８０点

とわかるわけです。これを表示させたいなら

ＰＲＩＮＴ　〝ＳＣＯＲＥ＝〟；ＳＣＯＲＥ

でできます。また，

ＳＣＯＲＥ＜０

となったら，

ＰＲＩＮＴ　〝ＧＡＭＥ　ＯＶＥＲ！〟

ＰＲＩＮＴ　〝アナタ　ノ　マケテ゛ス。〟

とやったりします。

マシン語における情報の記録

必要な情報を変数に記憶させる

ということは，ＢＡＳＩＣなら可能です。では，マシン語ならどうでしょうか？

マシン語だってＧＡＭＥを作ることは，あります。なにせ我々の共通の目標は，

オール・マシン語版

スペース・インベーダー

の制作ですね。とすると，プログラムの途中でさまざまな情報を記録しておく必要があります。

現在の得点

今までの最高点

面（シーン）

ビーム砲の残り数

⋮

いくらでも思いつきます。

それでは，これらの情報をマシン語ではどのように記憶させるのでしょう？

――**レジスタ**を使う。

結構です。我々は，初めてレジスタに接した時，

ＢＡＳＩＣにおける変数のようなもの

としてとらえました（「ＰＣ-8001　マシン語入門」第一巻・第２ブロック)。しかし，**レジスタの数には限り**があります。

ＢＣレジスタ＝現在の得点

ＤＥレジスタ＝最高点

⋮

とやっていったら，やがてレジスタが足りなくなってしまいます。これでは，必要な情報を記録させるどころか，肝心な

プログラミング

ができなくなってしまいますね？

――それなら**ＰＵＳＨ命令**を使ってレジスタの値を**スタック領域**に退避させたらいいさ。

というかもしれません。たしかに「ＰＣ-8001　マシン語入門」（第一巻）では，その方法を用いました。しかし，これではうまくいかないのです。

メモリに情報を

なぜなら，

ＰＵＳＨ　命令

ＰＯＰ　　命令

を使う時には，

ラスト・イン・ファスト・アウト

の原則があるからです（「ＰＣ-8001　マシン語入門」第１巻，P.122)。

第50図を御覧ください。レジスタが足りなくなり，

ビーム砲の数

最　高　点

得　点

の順に，その値を**スタック領域**に積んで行きました。すると，

ラスト・イン・ファスト・アウト

の原則により

得　点

最　高　点

ビーム砲の数

の順にしか，その値を取り出すことはできません。

仮にこの状態で，〝**ビーム砲の数**〟が必要となったとします。しかし，**スタック領域**からその値をすぐに取り出すことは，このままでは不可能です。

ＢＡＳＩＣと異なり，普通，マシン語で種々の情報を記憶させるには，

メ　モ　リ

を使います。たとえば,

得点………{ D000H
　　　　　　D001H

最高点……{ D002H
　　　　　　D003H

ビーム砲……D004H

のように,情報別に記憶させておく番地を決めておきます。それが,**2バイト分の情報量**なら**2バイト分の番地**を,また**1バイト分の情報量**なら**1バイト分の番地**を割り当てておきます。そしてたとえば,今,Aレジスタにビーム砲の数が入っていたとします。

Aレジスタ=02H

なら,

ビーム砲の残り数=2

というわけです。そしてこの情報を割り当てたメモリ(D004番地)にしまっておきたければ,

LD　(0D004H),A

とすればできます。逆に割り当てたメモリから情報を取り出したければ,

LD　A,(0D004H)

でAレジスタにビーム砲の数が入ります。

また,今,HLレジスタに得点が入っているとします。

HLレジスタ=0064H

なら,

現在の得点=100点

ということです。

(注) 16進数の64Hは,10進数で100です。

この値を割り当てたメモリにしまいたければ(D000番地とD001番地),

LD　(0D000H),HL

でできます。この時,

D000H=Lレジスタの値=64H

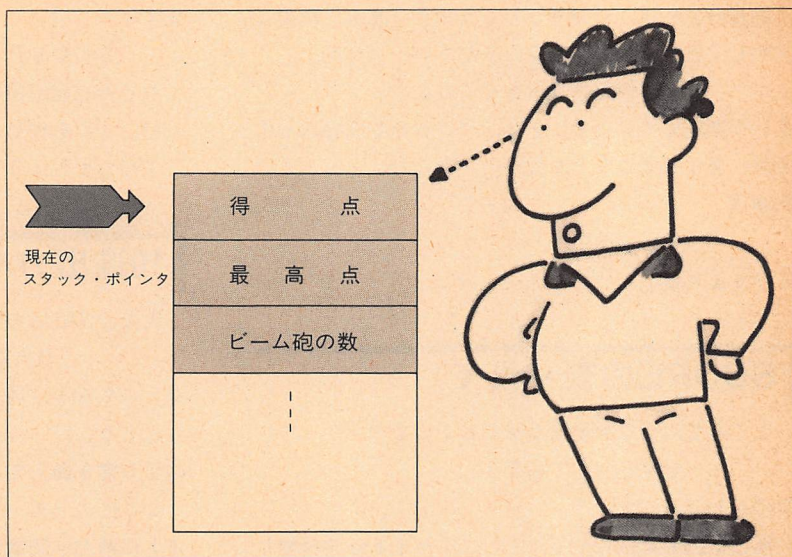
D001H=Hレジスタの値=00H

のようにしまわれることに御注意ください。

逆に割り当てたメモリから情報を取り出したければ,

LD　HL,(0D000H)

でHLレジスタに現在の得点が入ります。その結果

HL≧05DCH

第50図　ビーム砲の数を知りたいのだが

であれば,ビーム砲の数を一つ増やしてやろうということになります。

(注) 16進数の05DCHは,10進数で1500です。ちなみに私の作りました"スペース・インベーダー"では,1500点を越えると,**ビーム砲が一台追加**されることになっています。

得点表示に挑戦

以上のようにマシン語では,あらかじめ

メモリのある特定の番地

をあけておき,そこに

種々の情報を記憶させる

ということをしています。もう,お分りでしょう。まさにこのことのために

DS命令

は存在しています。

それでは,実例をお目にかけましょう。

〈チャレンジ〉

メモリの適当な番地(2バイト分)を得点記憶用のメモリに割り当て,そこに記憶させてある得点を表示させるプログラムを作りなさい。

問題の意味は，お分りでしょうか？　まあ，順番に説明して行きますから御安心ください。とにかく，

現在の得点を表示させてみよう

という問題です。このことは，マシン語でGAMEを作る際，必ず必要になりますから，あなたにとってこの問題は

必須事項

となりますね。

SCOREの割り当て

ところで，得点を記憶させておくのに

なぜ２バイト必要？

なのかお分りですか？

仮に得点を記憶させておくのに**１バイト分の領域**しか用意しなかったとします。すると，

００Ｈ～ＦＦＨ

までの得点を記憶できるわけです。これを10進数で表わせば，

０～２５５(点)

までの得点を記憶できるということです。

最高点＝２５５点

というのは，いかにも**少ない**ですね？　もし得点記憶用に**２バイトの領域**を確保したとすれば，

００００Ｈ～ＦＦＦＦＨ

までの得点を記憶できることになります。これは，10進数に換算すれば，

０～６５５３５(点)

までの得点を記憶できることになります。これなら実用上，十分ですね？

以上が，得点記憶用に２バイトの領域を確保した理由です（ちょっと過保護かな？）。

(注)　もし得点を10点単位にしか使わないなら，**１の位にダミーの０をつける**ことで，２バイトの数でも，

10～６５５３５０(点)

までの得点を表示させることができます。

それでは,〈チャレンジ〉の解法に移ります。まず得点記憶用のメモリを２バイト分用意します。

ＤＳ　２

でOKですね。ついでに**ラベル**もつけておきましょう。得点ですからSCOREというラベルをつけることにします。

ＳＣＯＲＥ：ＤＳ　２

これで得点記憶用のメモリが，２バイト分用意されました。これを何番地に割り当てるかは，あとで決めることにしましょう。

10進出力ルーチン

実際のGAMEの中では，最初この２バイトに

００００Ｈ

を入れておき（得点の初期値＝０），以後GAME の進行にともない，得点が加算されるたびにこの２バイトの値を**書き換え**ていきます。

ここでの〈チャレンジ〉では，**あらかじめこの２バイトに得点が記憶されている**との仮定のもとで，その値を10進数で表示するだけです。

それでは，その２バイトの値をレジスタに取り出しましょう。ＨＬレジスタを使うことにします。

ＬＤ　ＨＬ，(ＳＣＯＲＥ)

これでＨＬレジスタに，得点の２バイトの値が入りました。もちろん値は，**16進数**で入っています。あとは，これを表示するだけなのですが，実はこれが割と**やっかいな問題**です。というのは，

16進数──→10進数

の変換を実行してから，テレビ画面に表示しなければならないからです。

やがて我々が挑戦することになる

オール・マシン語版

スペース・インベーダー

では，

〝16進数→10進数変換ルーチン〟

を**作成する**ことになります。しかし現時点でそれをやっていると，

アセンブル・リストを理解する

という我々の当面の目標がボケてしまいます。そこでこの〈チャレンジ〉では，**ＲＯＭ内の〝10進出力ルーチン〟**を利用させてもらうことに致しましょう。ＲＯＭ内のサブルーチンの中には，我々の要求にピッタリなのがあります。

〈10進出力ルーチン〉
番　地：２Ｄ１３Ｈ
入力条件：
　ＨＬレジスタに出力したい２バイトの16進数を入れる。
機　能：
　ＨＬレジスタの値を10進数に変換した上でＣＲＴ画面に出力する。

得点を入れて

使い方は，これを見ただけでお分かりでしょう。すでにＨＬレジスタには，表示させたい値が入っていますから，あとはこの**サブルーチンをＣＡＬＬするだけ**です。

　　ＣＡＬＬ　２Ｄ１３Ｈ

これで所期の目的が達成されます。

以上までのプログラミングをまとめましょう。**第51図**が完成したプログラムです。ラベルを多用してありますが，中身はすべてお分りになるでしょう。**得点記憶用のメモリは，プログラムの一番最後**に割り当てることにしました。

　　Ｃ１０９Ｈ
　　Ｃ１０ＡＨ

の２バイトがそうです。ちなみに私の場合，ある程度大きなプログラムは，

- メイン・ルーチン
- サブルーチン群
- ＤＡＴＡ領域
- 各種情報記憶用

のようにメモリを割り当てることにしています。しかし，これはあくまでも**主観的な問題**ですから，あなたの好みで御自由にどうぞ。

プログラムができ上りましたので，いつものように

　　Ｓコマンド

を使って入力していきます。

　　Ｃ１００Ｈ～Ｃ１０８Ｈ

までの９バイトです。

　　Ｃ１０９Ｈ～Ｃ１０ＡＨ

の２バイトは，プログラムとして入力する必要はありません。

次に，

　　ＤＣ１００，Ｃ１０８↘

で確認です（**第52図**）。

さあ，ここでいつもなら，**Ｇコマンド**の登場，ということになるのですが，このプログラムでは**走らせる前に準備が必要**です。そうです。

　　Ｃ１０９Ｈ～Ｃ１０ＡＨ

の２バイトに，何か適当な

　　得　点

を入れておく必要があります。

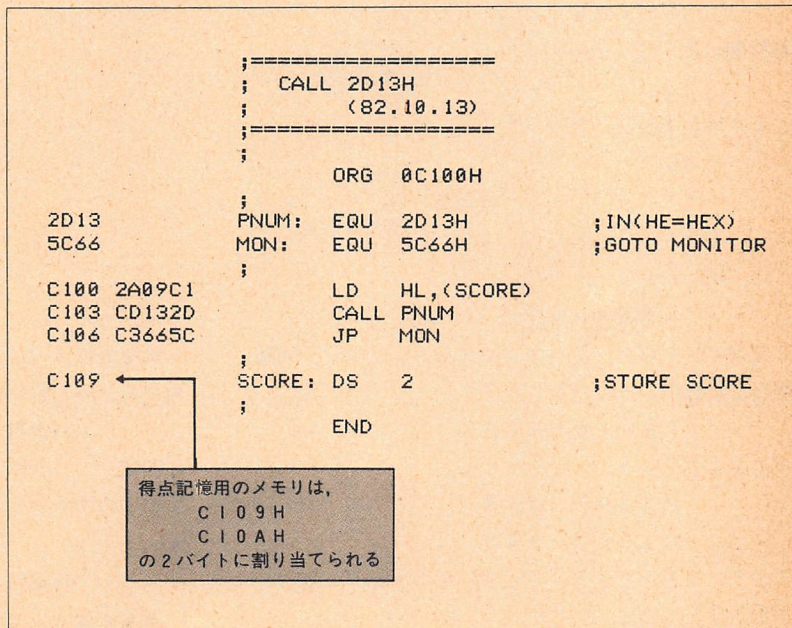
いろいろな16進数で

まずは簡単なところで

　　１　点

を表示させてみましょう。これを**２バイトの16進数**に変換すると，

　　０００１Ｈ

```
                      ;===================
                      ;  CALL 2D13H
                      ;       (82.10.13)
                      ;===================
                      ;
                              ORG   0C100H
                      ;
2D13                  PNUM:   EQU   2D13H          ;IN(HE=HEX)
5C66                  MON:    EQU   5C66H          ;GOTO MONITOR
                      ;
C100  2A09C1                  LD    HL,(SCORE)
C103  CD132D                  CALL  PNUM
C106  C3665C                  JP    MON
                      ;
C109                  SCORE:  DS    2              ;STORE SCORE
                      ;
                              END
```


第51図　得点表示プログラム

```
¥DC100,C108
C100 2A 09 C1 CD 13 2D C3 66 5C
¥
```

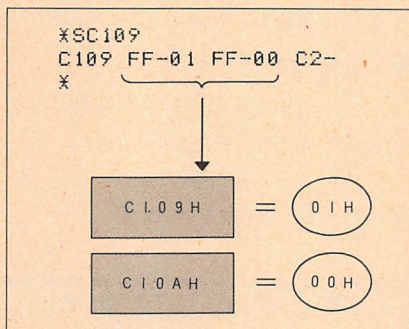
第52図　Dコマンドで確認

この**上位**と**下位**を入れ換え，

　01H，00H

の2バイトをC109H～C10AHに入れてやります。使うコマンドは，もちろん

　SC109H↘

第53図のとおりです。

第53図　0001(点)をセットする

　これで得点がセットされました。いよいよ，プログラムを走らせます。

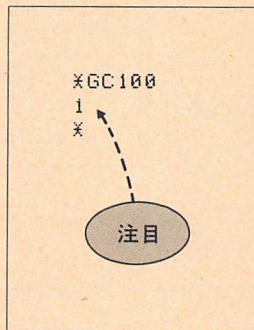
　GC100↘

第54図のように，見事(?)

　1

が表示されました。アッパレ，アッパレ！

```
¥GC100
1
¥
```

注目

第54図　やった！1点が表示された。

　1→0001H

では，何だかうまく

　10進数→16進数

の変換が行われたのかわかりません。もう少し，**16進数らしい数**でやってみましょう。

　10点＝000AH

は，どうでしょう。

　C109H＝0AH

　C10AH＝00H

のようにセットします。そして，Gコマンドで実行しましょう。みごと

　000AH＝10

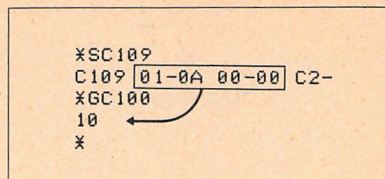
の変換が行われました(**第55図**)。

```
¥SC109
C109 01-0A 00-00 C2-
¥GC100
10
¥
```

第55図　もっと16進数らしく

　まだ物足りない？

　100＝0064H

では，いかがでしょう？**第56図**のようです。

```
¥SC100
C100 2A-64 09-00 C1-
¥GC100
49408
¥
```

第56図　100ならどうだ！

　まだ，まだ。それじゃ，目いっぱい大きく，

　FFFFH

でやってみましょうか？

　FFFFH＝65535

です。その計算の仕方は，ここでは説明しませんが，

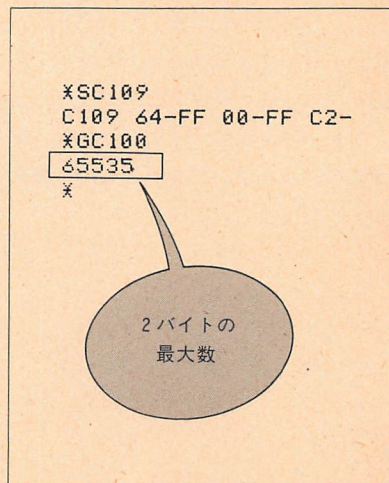
$$15\times16^3+15\times16^2+15\times16+15$$

または，

$$16^4-1$$

です。**第57図**のとおり，みごと

　65535

が表示されました。

目いっぱい大きく，FFFFHで。

　例をあげるのは，この辺でやめておきますが，その他いろいろな数で実験してみてください。

　16進数と**10進数**

の関係が，ハッキリとわかってくるでしょう。そうです。このプログラムは，〝得点を表示する〟だけでなく，

　16進数──→10進数

の変換を行ってくれるプログラムだったのです。

ワーク・エリア

以上で，第51図のプログラムの説明は終ります。マシン語では，**種々の情報**を

変　数―――――――――×

に記憶させるのでなく，

メ　モ　リ―――――――――○

に記憶させてプログラミングしていくのだということが，お分りいただけたことと思います。そして，そのために

DS命令

が存在するのだ，ということも納得いただけたでしょう。このために設けられたメモリ上の領域のことを，

ワーク・エリア

と読んでいます。

〈ワーク・エリア〉working area
　プログラム中，メモ的に使用するメモリのこと。

第6章

最後はEND

```
D780: 2B 10 FC AF 32 B9 E3 32 BA E3 26 0C 3A B2 E3 6F ; 07F3
D790: 22 BB E3 CD A1 D7 CD B2 D7 CD B8 D5 CD C7 D7 18 ; 0B38
D7A0: 42 CD 95 D9 E6 01 C8 3E 02 32 BA E3 3E 3E 32 BC ; 07A5
D7B0: E3 C9 3A B1 E3 06 FF 17 38 04 CB 18 18 F9 78 3F ; 077D
D7C0: 1F E6 0F 32 B0 E3 C9 11 A6 DF 2E 16 1A A7 C8 47 ; 074C
D7D0: 13 1A D5 F5 C5 E5 CD 60 D9 E1 C1 F1 2D 10 F4 D1 ; 0A3C
D7E0: 13 18 E9 06 05 11 DD DF 26 0C 3A BD E3 6F C5 E5 ; 0711
D7F0: CD 07 D8 E1 C1 78 E6 01 20 08 7B D6 18 5F 7A DE ; 07F5
D800: 00 57 2D 2D 10 E8 C9 06 0B C5 D5 E5 CD 72 D5 E1 ; 07F7
D810: D1 C1 7C C6 05 67 10 F1 C9 AF 32 C5 E3 21 16 02 ; 07CC
D820: CD 0C D9 06 4C 7E FE EE 20 05 3E FF 32 C5 E3 23 ; 07CD
D830: 10 F3 3A C5 E3 A7 C8 3E 03 32 AA E3 2E 16 06 03 ; 06A1
D840: C5 E5 3E 58 CD 60 D9 E1 E5 3E 3C CD 6E D8 E1 C1 ; 0A3B
D850: 2C 10 ED 06 05 C5 3E 21 D3 51 01 50 10 CD 15 DA ; 0599
D860: 3E 20 D3 51 01 50 60 CD 0D DA C1 10 E8 C9 4F CD ; 0785
D870: FD D8 EB 06 50 79 B6 77 23 10 FA C9 11 00 12 2A ; 06FF
D880: C6 E3 C3 B6 D9 11 00 25 2A C8 E3 C3 B6 D9 11 00 ; 0869
D890: 35 2A CA E3 C3 A9 D9 01 20 00 11 50 F3 21 94 E0 ; 075B
D8A0: ED B0 01 50 00 11 00 F3 21 B4 E0 ED B0 CD 7C D8 ; 0865
D8B0: 18 D3 21 12 0A CD C7 D8 21 12 1B CD C7 D8 21 12 ; 0681
D8C0: 2C CD C7 D8 21 12 3D 0E 04 11 05 E1 06 09 E5 D5 ; 05DA
D8D0: C5 CD 0C D9 C1 D1 1A 77 23 13 10 FA E1 2C 0D 20 ; 0714
D8E0: EB C9 01 19 50 CD 3A 09 21 18 01 22 5D EA 01 FF ; 05D1
D8F0: 00 CD F7 08 21 00 F8 22 5A EA C3 5A 04 E5 7D 07 : 06D5
```

きじめいれい　えんど……

ソース・プログラム

さあ，いよいよ

アセンブル・リストの見方

最終コーナーになります。最後の命令は，最後にふさわしく，

END命令

です。

このEND命令は，

アセンブルをSTOPさせる

のに用います。と言っても，なかなかハッキリ理解しにくいと思われますので，ここでも例をあげて説明することに致しましょう。

まず**第58図**を御覧ください。何の変哲もない

アセンブラのリスト

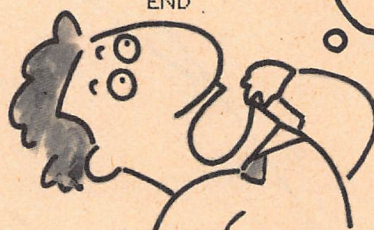
です。でも良く見ると，左側のところに

マシン語

が見当りませんね？　つまりこのリストは，純然たるアセンブリ言語のリストです。まだ

マシン語への変換

は行われていません。このリストからマシン語への変

```
;==================
;   END (82.10.18)
;==================
;
ORG 0C100H
;
NBASIC:EQU 81H
ATTRI:EQU 0EA5BH
;
LD A,20H
LD (ATTRI),A
JP NBASIC
;
END
```


第58図　ソース・リスト

換作業（アセンブル）を

手作業（ハンド・アセンブル）

で行うか，または，

アセンブラ

を使って行うかは，あなたの自由です。

第58図のように，まだマシン語に変換されていないプログラムを，

ソース・プログラム

また，そのリストを

ソース・リスト

と呼んでいます。ソース・プログラムのJISによる定義は，次のとおりです。

> **〈原始プログラム〉souce program**
> 原始言語（一つの言語であって，それから命令文が翻訳されるもの）で表された計算機プログラム（計算機による実行に適した形式で表されたプログラム）。

END命令の挿入

さて，第58図の

ソース・リスト

の一番最後を御覧ください。

ＥＮＤ命令

が見られますね？　この命令により，このプログラムの**アセンブル作業が，STOP**させられます。

それでは，実際にこのソース・プログラムをアセンブラにかけて，

アセンブル作業（マシン語への変換）

を行ってみましょう。あなたも，もしアセンブラをお持ちでしたら**実行**してみてください。

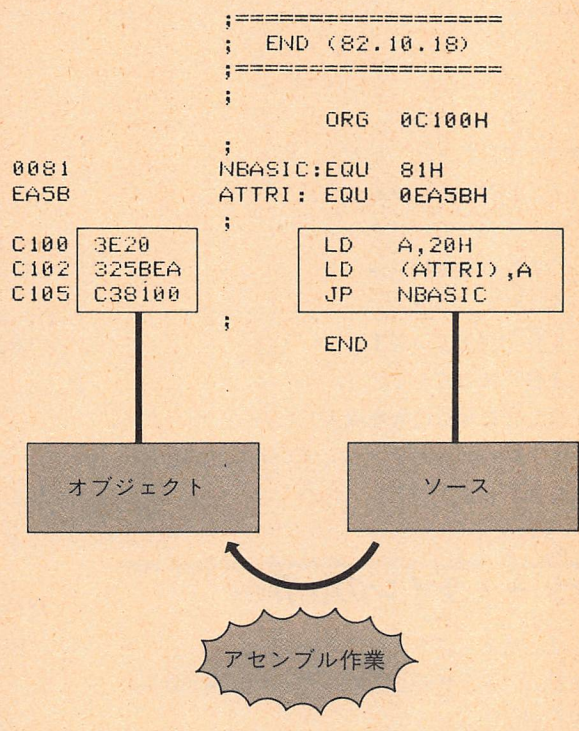
第59図が，アセンブルしたものです。今度は，ちゃんと左側の部分に

マシン語

が現われました。このできあがったマシン語のプログラムを，先程のソース・プログラムに対して

オブジェクト・プログラム

と呼んでいます。ＪＩＳによる定義は，次のとおりです。

> **〈目的プログラム〉object program**
> 原始言語から目的言語（一つの言語であって，それへ命令文が翻訳されるもの）に翻訳された計算機プログラム。

第59図　アセンブル作業の実行

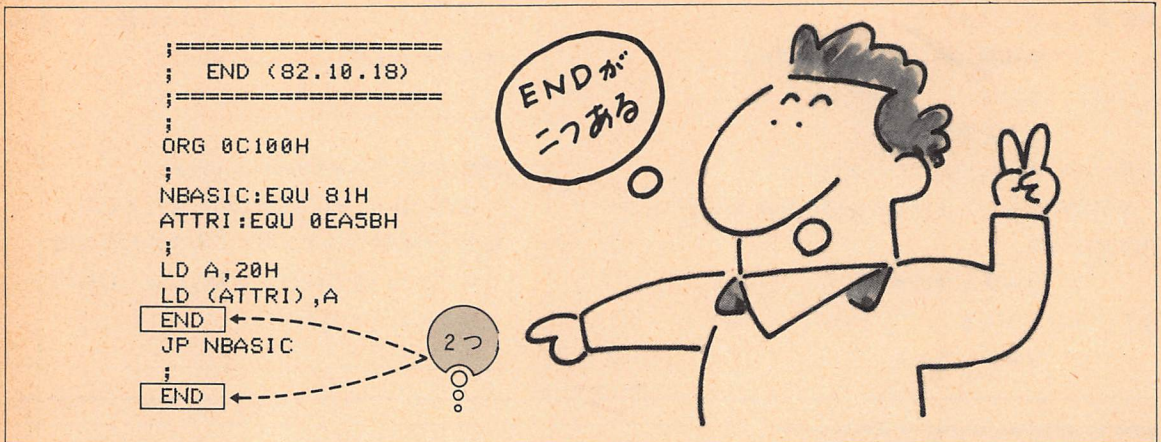
さて，第58図のソース・プログラムから，第59図のオブジェクト・プログラムへの

アセンブル過程

を見ても，あまり

ＥＮＤ命令の役割り

がピンとこないかもしれません。そこで，次のようなことをしたらどうでしょう？

第60図を御覧ください。これは，アセンブラのソース・プログラムです。そして，第58図のリストと比べてみてください。中身は，

第60図　END命令を途中に挿入する

まったく同じ

プログラムです。でも良く見ると，何と

「ややっ！　ENDが二つある！」

そうです。第60図のプログラムには，

END命令が二つ

入っているのです。

LD　(ATTRI), A

JP　NBASIC

の間に，新たに**END命令を挿入**したわけです。さあ，このソース・プログラムをアセンブルすると，どういうことになるでしょうか？

アンダー・ラインの世界へ

それでは，アセンブラにかけてみましょう。

第61図が，アセンブルしたものです。第59図のリストと比較してみてください。

C100H～C104H

までしかアセンブルされていないことがわかります。

C105H～C107H

については，アセンブルされていません。

いかがですか？　これで

END命令の意味

が，大部ハッキリしてきたのではないでしょうか？

END命令は，**同一プログラム内**に

複数個置くことが可能

です。しかし，アセンブラは

最初に出会ったEND命令のところで

アセンブル作業を中断

```
                    ;===================
                    ;  END (82.10.18)
                    ;===================
                    ;
                           ORG   0C100H
                    ;
0081                NBASIC:EQU   81H
EA5B                ATTRI: EQU   0EA5BH
                    ;
C100  3E20                 LD    A,20H
C102  325BEA               LD    (ATTRI),A
                           END
```

第61図　ENDでSTOP

してしまいます。なぜなら

END命令は，アセンブル作業を

STOPする命令

だからです。

(注)　END命令は，同一プログラム内に何個置いても構いません。しかし，**最低一個**は**プログラムの最後**に置かなければなりません。なぜなら，END命令がないと，アセンブラは，**どこでアセンブル作業をやめてよいのかわからなくなる**からです。しかし，なかにはEND命令不要のアセンブラもあります。まさに，アセンブラにもさまざまなものがあるわけです。

以上で**END命令の機能**についての話しは，おしまいです。ここで先に進んでも良いのですが，まだ一つ，気になることが残っていましたね？

そうです。せっかく第58図で取り上げたプログラムを，我々はまだ走らせていませんでした。次に進む前にせっかくですから走らせておきましょう。それには，

第59図のマシン語を入力すると良いでしょう。間違っても，**第61図のマシン語を入力してはいけません**。なぜなら，第61図のプログラムは

途中までしかアセンブルされていない！

からです。プログラムを入力しましたら，

DC100，C107↘

で確認です（**第62図**）。そして

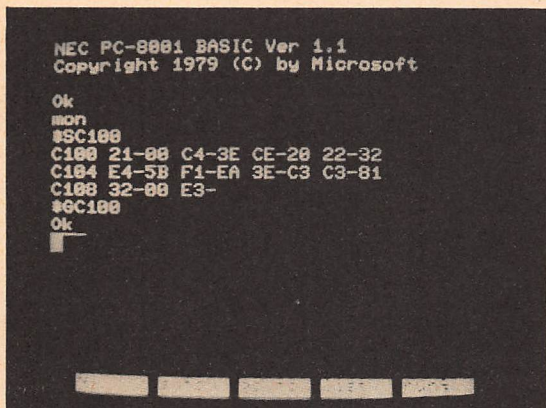
GC100↘

でスタートさせてください（**写真1**）。

```
*DC100,C107
C100 3E 20 32 5B EA C3 81 00
*
```

第62図　Dコマンドで確認

```
NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft

Ok
mon
*SC100
C100 21-00 C4-3E CE-20 22-32
C104 E4-5B F1-EA 3E-C3 C3-81
C108 32-00 E3-
*GC100
Ok
```

《写真１》ＧＣ１００でスタート

何が起こりましたか？

ＢＡＳＩＣに戻ってしまいましたね。え？　何か変だ？　そういえば何か変ですね。

PC-8001

とキーインしてみてください。ややっ！

アンダー・ライン！

アンダー・ラインが現われました（**写真２**）。他にも何か適当にキーをたたいてみてください。キーをたたくたびに，文字の下にアンダー・ラインが現われますね（**写真３**）？　そうです。第58図のプログラムは，

アンダー・ラインを出す

ためのプログラムだったのです。

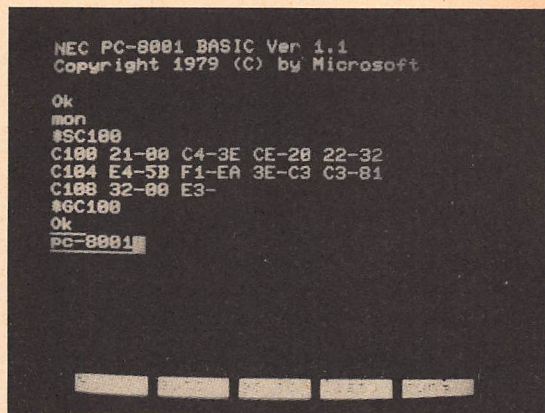
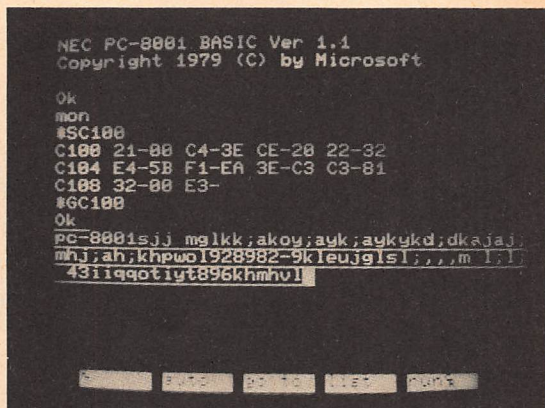
なお，これを**ノーマルの状態**に戻したければ，

COLOR　0↘

を実行します。その後，またアンダー・ラインを出したくなったら，

《写真２》ＰＣ－８００１とキーイン

```
NEC PC-8001 BASIC Ver 1.1
Copyright 1979 (C) by Microsoft

Ok
mon
*SC100
C100 21-00 C4-3E CE-20 22-32
C104 E4-5B F1-EA 3E-C3 C3-81
C108 32-00 E3-
*GC100
Ok
pc-8001sjj mglkk;akoy;ayk;aykykd;dkajaj;
mhj;ah;khpwol928982-9kleujglsl;,,,m l;l
 43iiqqotiyt896khmhvl
```

《写真３》適当に２～３行キーイン

MON↘

GC100↘

でOKです。

擬似命令

以上をもちまして，

アセンブル・リスト

についての話しは，すべておしまいです。最後にそれらをまとめておきましょう。

今まで，我々はアセンブラに使う

命　令

をいろいろ見てきました。その際，単に

××命令

と呼んでいました。しかし，正確にはそれらの命令は，

擬似命令

とか，

アセンブラ指示語

とか呼ばれています。

〈擬似命令〉 pseudo instruction

アセンブラに対する命令。見かけ上は，ＣＰＵに対する命令のような形をとるが，アセンブルはされない。

以下に，このブロックで見てきた

擬似命令＝アセンブラ指示語

をまとめておきます。

① ＯＲＧ (origin)

ロケーション・カウンタの初期化。以下，この命令で指定した番地からアセンブルされていきます。

② ＥＱＵ (equate)

ラベルの定義。指定したラベルに番地を与えます。

③ ＤＢ (define byte)

１バイトのデータの定義。

④ ＤＣ (define character)

文字データの定義。

⑤ ＤＷ (define word)

２バイトのデータの定義。マシン語に変換されるとき，データの上位バイトと下位バイトが逆転される。80系のＣＰＵにとっては，きわめて都合の良い命令。

⑥ ＤＳ (define strage)

指定された大きさの領域を確保する。

⑦ ＥＮＤ

アセンブラに対し，アセンブル作業の停止を指示する命令。

〈リビング・ルーム〉

—アセンブラのいろいろ—

本ブロックの各章で見てきましたように，ひとくちに**アセンブラ**といっても，それこそいろいろなタイプがあります。もっとも単純で使いやすいタイプは，いわゆる

オン・メモリ型

と呼ばれるもので，

テキスト・エディタ
アセンブラ
ソース・プログラム
オブジェクト・プログラム

がすべて**メモリ上**にあるものです。

テキスト・エディタは，ソース・プログラム（アセンブリ言語で書かれる）の

作成・訂正・挿入・削除

に使います。オン・メモリ型では，ソース・プログラムはそのままメモリ上に置かれます。この時，

ＡＳＣＩＩ形式

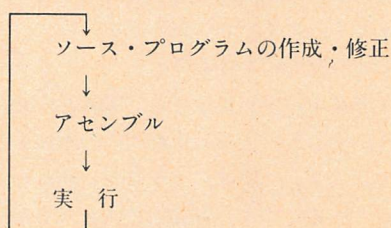
で置かれるタイプと，**中間言語**に圧縮されて置かれるタイプの二種類があります。中間言語化されるタイプでは，あとでリストを取る際に逆変換ルーチンを持たなければならない等，システムが大きくなるため，アセンブラ用のエディタでは少ないようです。

メモリ上に配置されたソース・プログラムは，アセンブラによりオブジェクト・プログラム（マシン語）に変換されます。オン・メモリ型では，このオブジェクト・プログラムもメモリ上に配置されます。したがって，

アセンブルされたマシン語は
Ｇコマンドで即実行可能

です。

走らせた結果バグがあった場合，オン・メモリ型ではソース・プログラムがメモリ上に残っていますから，すぐテキスト・エディタを使ってソース・プログラムの修正ができます。

以上のようにオン・メモリ型のアセンブラでは，

を，きわめて効率良く行うことができます。ハンド・アセンブルばかりしてきた人が，オン・メモリ型のアセンブラを使うと，

「これが，マシン語のプログラミング？」

と驚くことでしょう。

それでは，なぜ他のアセンブラでも，こんなに

便利なオン・メモリ型にしないのでしょう？
——それは，**メモリ容量**のためです。
考えてみればすぐにわかるように，オン・メモリ型では，

テキスト・エディタ
アセンブラ
ソース・プログラム
オブジェクト・プログラム

が同時にメモリ上にあるため，

アセンブル可能な量

が小さくなってしまうのです。たとえば

テキスト・エディタ＋アセンブラ

の**システムの部分**だけで**4～8Kバイト位**は必要です。他にラベル・テーブル等のワーク・エリアも大量に必要です。ソース・プログラムは，オブジェクトの数倍必要です。したがいまして，32KシステムのPC-8001でも，オン・メモリ型のアセンブラでは，

約2Kバイト

程度のマシン語しか作れません。ちなみに，オール・マシン語版スペース・インベーダーは，5Kバイト以上ありますから，オン・メモリ型のアセンブラでは作れないことになります。もちろん，何回かに分けてアセンブルすれば，できないことはないのですが，手間は倍以上かかります。

作製可能なマシン語の量を増やすため，いろいろなタイプのアセンブラが開発されています。

一つは，**ROMタイプ**のもの。

これは，PC-8001の空きエリアである

6000H～7FFFH

に，ROMを挿入することによりメモリを増設するものです。この部分に

テキスト・エディタ＋アセンブラ

が移動しますから，作成可能なマシン語も

約4Kバイト

位になります。長所は，スイッチ・オンですぐ使えること，また機能面もオン・メモリ型と同じと考えられますから，効率良くマシン語を作成することができます。

ところで大きなマシン語が作れない理由は，

テキスト・エディタ
アセンブラ
ソース・プログラム
オブジェクト

が，**同時にメモリ上に同居している**からです。したがって，これらの一部をメモリから追い出してやれば，大きなマシン語を作ることができます。

この考え方によるもので，最も簡単なものは，**オブジェクト（マシン語）をテープ上に出力するタイプ**のものです。このタイプのものは，アセンブルと同時にマシン語をテープにSAVEします。そしてプログラムを実行する時は，そのテープをLOADします。そのため，

アセンブル──→実行

に時間がかかりますが，ROMタイプと同じ位の大きさのマシン語が作れます。

DISKを利用するアセンブラは，面白いのが沢山あります。なかには，大型機並のOSを備えていて，**マクロ命令**が使え，**リンケージ・エディタ**によりオブジェクトを作成するという大掛りなものまであります。

DISK版による最も基本的なアセンブルの過程を御紹介しますと，

テキスト・エディタのLOAD
テキスト（ソース・プログラム）作成
テキストのSAVE
アセンブラのLOAD
テキストのLOAD
アセンブル
リロケータブル・ユニットのSAVE
リンケージ・エディタのLOAD
必要なユニットのLOAD
連結編集（オブジェクト完成）
オブジェクトのSAVE
ローダのLOAD
オブジェクトのLOAD
↓実行

以上のように**ファイルの入出力の回数が非常に多く**なります。このため，大型機並のしっかりした**JOB管理プログラム**が要求されます。

このようにDISK版のアセンブラは，複雑になりがちですが，かなり大きなプログラムを作成することが可能です。

入門編から実務応用編まで

幅広いマイコン情報を掲載

DEMPAマイコンテープ・BOOKsはあなたのマイコン知識を豊富にする！

DEMPAマイコンテープ・BOOKs

最新

新刊

←PC-6001活用研究

実践プログラム集

PC-6001活用研究

プログラミングの基礎からマシン語の応用まで

●定価 1,300円　送料250円

8種類の実用プログラムから15種類の最新ゲームのBASICプログラムとマシン語プログラムまでを同時掲載。マシン語モニタ、逆アセンブラ、からマシン語応用編を紹介。PC-6001利用者の必読書。

←OA導入80のポイント

OFFICE AUTOMATION GUIDE

OA導入80のポイント

OA時代を勝ち抜く法

マネージアカデミー所長 安田 賀計著

貴方を変えるOAの一般常識80。

定価 1,200円　送料250円

第三の波、OA革命が到来した。OA機器は職場を変え、オフィスロボットはあなたの仕事を変える。ビジネスマンのためのOA一般常識80。

←マイコンBASIC講座3

マイコンBASIC講座③

プログラム・マスター編

マイコンBASICマガジン別冊

●定価 1,400円　送料300円

パソコン利用者のプログラム活用を高めるために全人気機種の画面表示をとりあげSTR$・CHR$などを実例解説。プログラムリストも短かく、多くの例題を習得してプログラムをマスターして下さい。

←PC-8001活用マニュアル

PCファンのための必読保存版

月刊マイコン別冊

PC-8001活用マニュアル

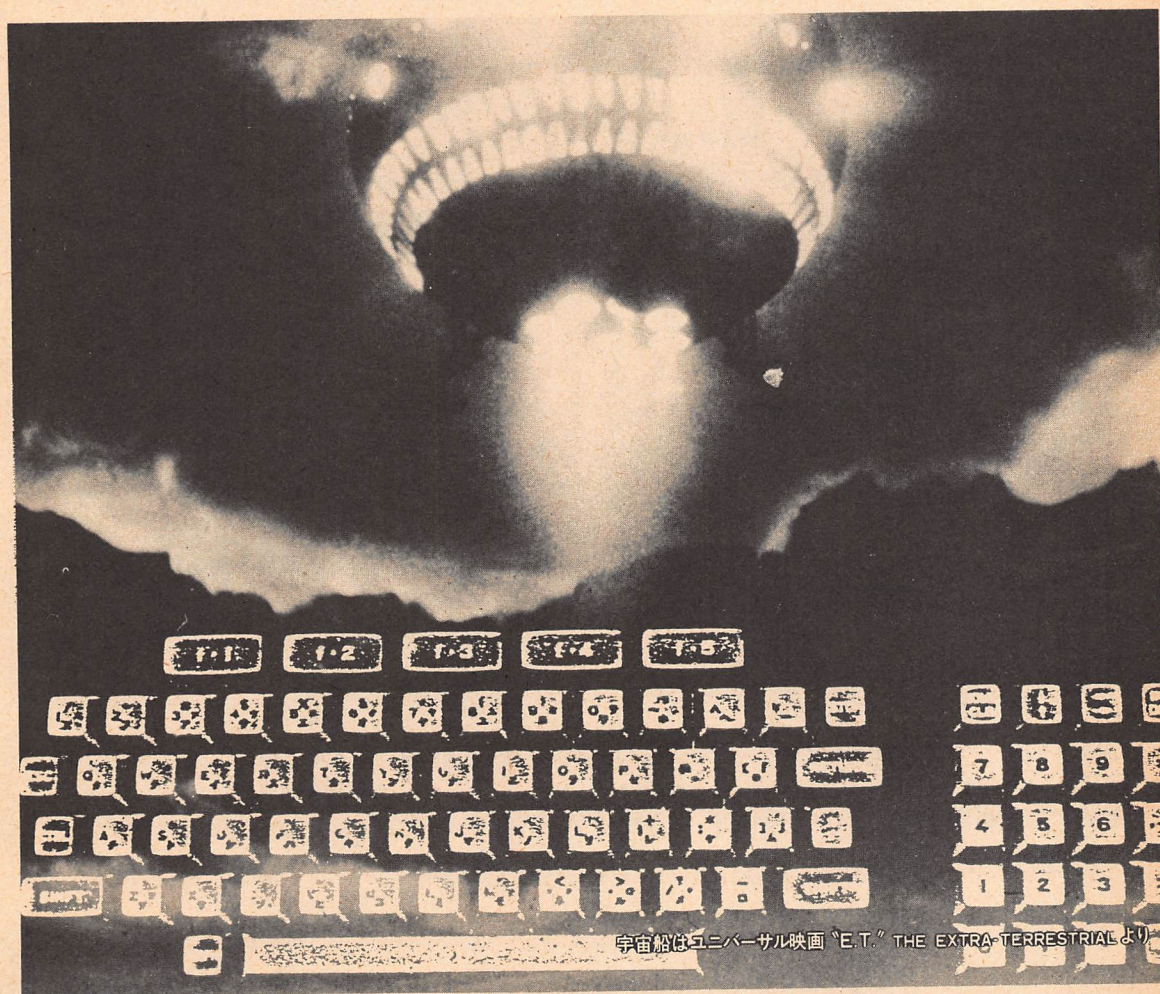
好評のプログラム20本一挙掲載！

●定価 1,200円　送料250円

プログラム入門から各種アプリケーションまでPC-8001関連情報を集大成。好評のプログラム20本を一挙掲載。

第2ブロック

USR関数への招待

宇宙船はユニバーサル映画〝E.T.″THE EXTRA-TERRESTRIALより

〈はじめに〉

ＵＳＲ関数――この言葉を聞くと，どんな感じがしますか？　**嫌悪感を抱く人**，**拒絶反応を起こす人**等々さまざまでしょう。もしあなたもこういった感じを持たれるなら，あなたも

ＵＳＲ関数コンプレックス病

の症状が見られます。

確かにマニュアルを見ますと，「ＵＳＲ関数」の項は読みにくく，親しみにくいかもしれません。しかし，

BASIC＋マシン語

を用いて**効率の良いプログラム**を作ろうとすると，どうしても

ＵＳＲ関数の知識が不可欠

となります。かつ，ＵＳＲ関数くらいチョチョイのチョイでないと，ＰＣ-8001については

マシン語の中級者になった

とはいえないでしょう。

かくて我々は，ＵＳＲ関数のマスターに向けて，その努力の第一歩を踏み出すことになります。

ただし，――。

私は，その努力が少しでも軽く，また甘い

蜜の道（ハニー・ロード）

〔カッコイイ！〕

であるよう，考えに考え（休む時は休み），最高の構成でその**教程**を準備しました（実際は，それ程オオゲサではありません）。

しかし，――。

興味深く，かつ（少しでも）楽しみながら読み進められるよう，その構成は複雑をきわめています。話の途中で，話題がアチコチに飛びます。またきれいサッパリ忘れ去った頃，元の話題に戻って来たりします。その間，読者は**混乱の極致**に陥ることでしょう。しかし，これは楽しみながらマスターするための混乱ですから，どうぞ安心して心やすく**パニック状態**に陥ってください。

でも，――。

その中ではたくさんの具体例がでてきます。それについては絶対に手を抜かず，必ず**ご自分の手で実験**してみてください。いつのまにかそこに出てくるマシン語の命令をマスターできるでしょう。

話しの構成上，USR関数マスターのための道具として，最初に〝ミニ・レジスタ表示プログラム〟の話題が続きます。そして，その使い方を理解するために，**リロケートの問題**が登場します。それでも我慢して読み続けていくと，本ブロック終了時には，

基本的なＵＳＲ関数の使い方

はマスターされることでしょう。ただし，より複雑な使い方については，次ブロックで扱うことになります。

以上のように，ここしばらくは，

スパゲッティ風

支離滅裂な構成

が続きます。しかし，それは今のあなたにとって，**最高の教程**なのです。すなわち

ＵＳＲ関数のマスターには

〝王　道〟があった！

のです。

第7章

ミニ・レジスタ表示プログラム

```
D900: 5F 16 00 21 75 06 19 5E 23 56 E1 C9 CD FD D8 6C ; 06B9
D910: 26 00 19 C9 CD FD D8 EB 01 50 00 09 C9 E5 CD 27 ; 0791
D920: D9 E1 01 78 00 09 13 1A A7 C8 77 13 23 18 F8 C5 ; 065A
D930: E5 CD 3B D9 E1 01 78 00 09 C1 13 AF 77 23 1A 77 ; 06D7
D940: 04 13 23 1A A7 C8 70 23 77 04 13 23 18 F5 E5 C5 ; 05BE
D950: F5 CD 0C D9 F1 77 23 10 FC C1 E1 2C 0D 20 EF C9 ; 08F1
D960: F5 CD 14 D9 AF 77 F1 23 77 01 50 13 23 71 23 77 ; 06F2
D970: 10 FA C9 CD 85 D9 10 FB C9 11 FF FF 1D C2 7C D9 ; 0A15
D980: 15 C2 7C D9 C9 1E FF 1D 20 FD C9 CD 95 D9 90 28 ; 0908
D990: 02 30 FB 80 C9 E5 2A CB E3 7C 85 6F 7C C6 03 67 ; 084F
D9A0: 7D CE 00 6F 22 CB E3 E1 C9 C5 E5 EB CD 0C D9 EB ; 0A66
D9B0: E1 D5 26 00 18 15 C5 E5 EB CD 0C D9 EB E1 D5 01 ; 08F2
D9C0: 10 27 CD EF D9 01 E8 03 CD EF D9 01 64 00 CD EF ; 086E
D9D0: D9 0E 0A CD EF D9 7D C6 30 12 E1 7E FE 30 20 05 ; 07BD
D9E0: 36 20 23 18 F6 1A FE 20 20 03 3E 30 12 C1 C9 C5 ; 05B1
D9F0: AF 3E 30 ED 42 38 03 3C 18 F9 C1 09 12 13 C9 F5 ; 0681
DA00: 3E FF CD 1E DA 3D 20 FA 0D 20 F5 F1 C9 CD 1E DA ; 08FA
DA10: 04 0D 20 F9 C9 CD 1E DA 05 05 0D 20 F8 C9 D5 50 ; 06D5
DA20: 3A 67 EA CB EF D3 40 15 20 FB 50 3A 67 EA CB AF ; 08DD
DA30: D3 40 15 20 FB D1 C9 7E A7 C8 47 23 4E 07 30 05 ; 06BE
DA40: CD 58 DA 18 03 CD 8D DA 3A 67 EA CB AF D3 40 06 ; 086C
DA50: 10 CD 85 D9 10 FB 18 DF E5 D5 CD 64 DA D1 0D 20 ; 0900
DA60: F8 E1 23 C9 EB 50 3A 67 EA CB AF D3 40 2B 7C A7 ; 0966
DA70: 28 15 15 20 F8 50 3A 67 EA CB AF D3 40 2B 7C A7 ; 0720
```

ふたつ　の　みに　………

奇妙な条件

さらに次へのステップを目指して，**第2ブロック**のスタートです。

まず，**第63図**を御覧ください。この長いマシン語のリスト，どこかで見たことありませんか？

そうです。本書の最初の図である第1図と同じものです。ところで，このリストのマシン語，まだ走らせていませんでしたね？

実は，この第63図のリスト，

非常に重要なプログラム

が書かれているのです。それは，今後あなたが本書のマシン語を実験したり，新しいプログラムを作成していく上で，たびたび使用していくことになるでしょう。

しかし，このプログラムの使い方，**ちょっと取っつきにくい面**があります。そこで例をあげながら使い方を説明していくことに致します。この機会に，良くマスターしておいてください。

まず，次の〈チャレンジ〉を考えてみてください。

〈チャレンジ〉

次の条件にしたがって，プログラムを完成させなさい。

① スタック領域に0081Hをセットする。
② 1バイトの領域を確保する。
③ そこに70Hのデータをセットする。
④ その領域からデータを取り出し，Bレジスタにセットする。
⑤ Cレジスタに10（10進数）をセットする。
⑥ BCレジスタの値をDEレジスタに移す。
⑦ DEレジスタの上位バイトと下位バイトを逆転させる。
⑧ DEレジスタの値をHLレジスタに移す。
⑨ Lレジスタの値を＋2する。
⑩ HLレジスタの示す番地にジャンプさせる。

```
                 ;===================================
                 ;  MINI REGISTER DISPLAY
                 ;    (81年 3月 18日):BY K.TSUKAGOSHI
                 ;===================================
                 ;
                        ORG   0D000H
                 ;
0257             OCRT:  EQU   257H            ;OUT CRT
5C66             GMON:  EQU   5C66H           ;GOTO MONITOR
5EC0             PRHL:  EQU   5EC0H           ;PRINT HL
5FCA             CRLF:  EQU   5FCAH
5FD4             PSPC:  EQU   5FD4H           ;PRINT SPACE
                 ;
D000 FDE5        MAIN:  PUSH IY               ;REGISTER STORE FOR DISPLAY
D002 DDE5               PUSH IX
D004 E5                 PUSH HL
D005 D5                 PUSH DE
D006 C5                 PUSH BC
D007 F5                 PUSH AF
                 ;
D008 CDCA5F             CALL CRLF             ;CARRIAGE LINE FEED,PREPRA FOR DISPLAY
                 ;
D00B 0625               LD   B,37             ;'AF-SP'=37 CHARACTER
D00D 210000             LD   HL,0             ;LET HL=SP
D010 39                 ADD  HL,SP
D011 2B                 DEC  HL
D012 2B                 DEC  HL
D013 F9                 LD   SP,HL
D014 E1                 POP  HL
D015 113500             LD   DE,53            ;HL=ADR(DATA)
D018 19                 ADD  HL,DE
                 ;
D019 7E          MA1:   LD   A,(HL)           ;PRINT 'AF-SP'
D01A CD5702             CALL OCRT
D01D 23                 INC  HL
D01E 10F9               DJNZ MA1
                 ;
D020 CDCA5F             CALL CRLF
                 ;
D023 0606               LD   B,6              ;6 REGISTERS
D025 E1          MA2:   POP  HL
D026 CDC05E             CALL PRHL
D029 CDD45F             CALL PSPC
D02C 10F7               DJNZ MA2
                 ;
D02E E1                 POP  HL               ;PRINT PC
D02F 2B                 DEC  HL
D030 CDC05E             CALL PRHL
D033 CDD45F             CALL PSPC
                 ;
D036 210000             LD   HL,0             ;PRINT SP
D039 39                 ADD  HL,SP
D03A CDC05E             CALL PRHL
                 ;
D03D C3665C             JP   GMON             ;GOTO MONITOR
                 ;
D040 41462020    DATA:  DB   'AF    BC    DE    HL
D044 20424320
D048 20204445
D04C 20202048
D050 4C202020

D054 49582020           DB   'IX    IY    PC    SP'
D058 20495920
D05C 20205043
D060 20202053
D064 50
                 ;
                        END
```

第63図　どこかで見たリスト

何やらややっこしいことが書いてあります。いったい何をやらかそうとしているのでしょう？　今は，あまり中身のことは考えず，とにかくこの〈チャレンジ〉の条件を満たすプログラムを完成させてみてください。何が起こるかは，秘密です。

Aレジスタを中継基地に

プログラムそのものは，簡単です。なお，あらかじめ予告しておきますが，以下の作業の中で，わざと

ミ　ス

をおかします。もし気がつきましても，わざと知らないふりをしていてください。

まず最初に②番以降を考えていきます。②，③は，

〝1バイトの領域を確保し，
そこに70Hをセット〟

することです。これは，第1ブロックでやりましたように，**DB命令**を使えばできます。

```
DB  70H
```

ただ，これだけです。ついでに，この番地にラベルをつけておきましょう。DATAという名前をつけることにします。

```
DATA:DB  70H
```

これでOKですね。

次に④の

「その領域からデータを取り出し，Bレジスタにセットする」

を行います。「その領域」とは，メモリ上にあります。したがって，この部分は，

あるメモリから1バイトのDATAを
取り出し，Bレジスタにセットする

ということになります（**第64図**）。

ところで，**メモリからデータを取り出し，Bレジスタにセットする**という命令はありません。そこで一度，データをAレジスタに取り出し，その後Bレジスタに移してやる，という方法をとることになります。プログラムは，

```
LD  A,(DATA)
LD  B,A
```

でOKです。

(注)　付録の〝Z—80活用表〟を御覧ください。そして，

```
LD  B,(nn)
```

の命令がないことを御確認ください。

エレガントなスタック命令

次です。⑤の，

「Cレジスタに10（10進数）をセットする」

は，一つの命令でできます。

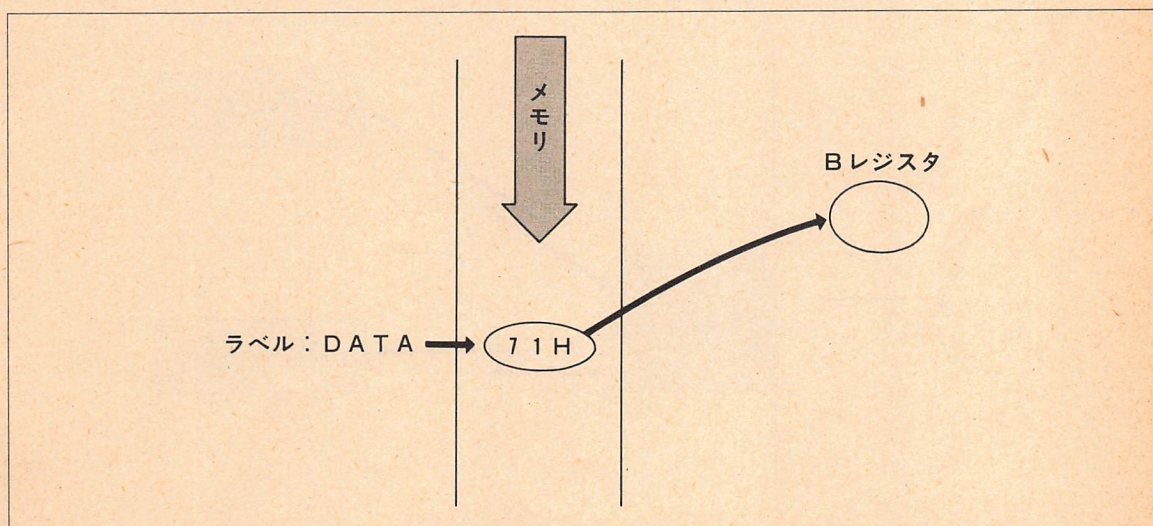
10（10進数）＝A（16進数）

であることに注意すれば，

```
LD  C,0AH
```

または，

```
LD  C,10
```

でOKです。

第64図　メモリ──→Bレジスタ

次に⑥の
「ＢＣレジスタの値をＤＥレジスタに移す。」
については，二つの方法を御紹介致しましょう。
一つは，オーソドックスにＬＤ命令を使い，

ＬＤ　Ｄ，Ｂ
ＬＤ　Ｅ，Ｃ

でできます。今一つは，もっとポピュラーな方法で，**スタック領域**を使うものです。その基本は，次のとおりです。

> **〈レジスタ・ペアの値を移す〉**
> (レジスタ・ペア１)⇐(レジスタ・ペア２)
> のように値を移したいときは，
> PUSH　(レジスタ・ペア２)
> POP　(レジスタ・ペア１)

いま我々のやりたいことは，

ＤＥ←─ＢＣ

のように値を移したいのですから，上の方法にしたがえば，

PUSH　ＢＣ
POP　ＤＥ

で実行できます。どうですか？　ＬＤ命令を二回使うよりは，エレガントでしょう。
ついでになぜこれで

ＤＥ←─ＢＣ

のように値が移されるか考えておきましょう。

まず，

PUSH　ＢＣ

でＢＣレジスタの値が，**スタック領域のトップ**に積まれます。次いで

POP　ＤＥ

を実行することで，スタック領域のトップにあったＢＣレジスタの値が，ＤＥレジスタに代入されることになります（**第65図**）。なおこの結果は

ＢＣレジスタ＝ＤＥレジスタ

となります。

(注)　PUSH　ＤＥ
POP　ＢＣ
とやっても，
ＢＣレジスタ＝ＤＥレジスタ
となります。しかし，この場合は
最初のＤＥレジスタの値
に統一されます。

LOADのいろいろ

続いて⑦の
「ＤＥレジスタの上位バイトと下位バイトを逆転させる」
です。その意味は

(上位バイト)　(下位バイト)
Ｄレジスタ←→Ｅレジスタ

のように，**Ｄレジスタの値とＥレジスタの値を交換**しなさい，というものです。

第65図　スタック領域を中継に

さて，これについては一回の命令ではできません。結局，こまめにＬＤ命令を使って実現させます。その場合でも，いきなり

ＬＤ　Ｄ，Ｅ

のようにやると，

Ｄレジスタ←──Ｅレジスタ

のように値は移りますが，**Ｄレジスタの値が消えてしまいます**。そのため

Ａレジスタ

を登場させ，ここを中継に値を移してやります。

ＬＤ　Ａ，Ｄ
ＬＤ　Ｄ，Ｅ
ＬＤ　Ｅ，Ａ

でできます。**第66図**を見ながら，考えてみてください。

第66図　Ａレジスタを使って

次の⑧

「ＤＥレジスタの値をＨＬレジスタに移す」

のは簡単です。三通りの方法を御紹介致しましょう。

Ⓐ　**ＬＤ命令を使う**

もっとも初歩的な方法です。

ＬＤ　Ｈ，Ｄ
ＬＤ　Ｌ，Ｅ

2バイトのマシン語で実現できました。

Ⓑ　**スタック領域を用いる**

この方法は，いま覚えたばかりですから，多くの人がこの方法で実行したのではないでしょうか？

ＰＵＳＨ　ＤＥ
ＰＯＰ　　ＨＬ

やはり**2バイトのマシン語**で実現できました。

Ⓒ　**交換命令を用いる**

ＤＥレジスタとＨＬレジスタは，その値を交換する命令がありました。

ＥＸ　ＤＥ，ＨＬ

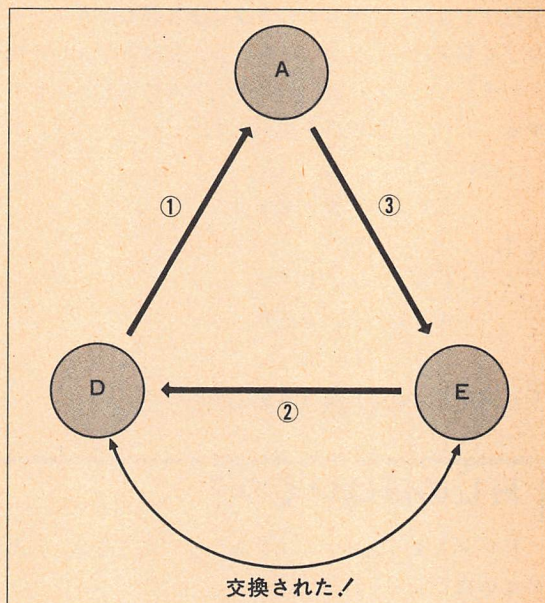
これでおしまいです。わずか**1バイトのマシン語**で実現できました。

加算命令の注意

続いて⑨

「レジスタの値を＋２する」

です。これについては，**二通りの方法**を御紹介致します。

最初は素直に，

〝＋２するのだから**ＡＤＤ命令**を使おう〟

というものです。**加算は加算命令で**，ときわめて素直な発想です。ただし，加算命令を使うときは注意がありまして，

加算命令はＡレジスタのみ可能

であったことを思い出してください。すなわち

ＡＤＤ　Ｌ，０２Ｈ

とやりたいところですが，こういう命令は存在しませんから，

ＡＤＤ　Ａ，０２Ｈ

を利用して行うことになります。

このことをまとめておきますと，次のようになります。

〈ＡＤＤ命令の利用法〉

①　**Ａレジスタの場合**

ＡＤＤ　Ａ，××
（↑加える数）

②　**Ａレジスタ以外の場合**

（Ｘレジスタとすると）

ＬＤ　　Ａ，Ｘ
ＡＤＤ　Ａ，××
ＬＤ　　Ｘ，Ａ

このように，Ａレジスタ以外のレジスタで加算命令

を行うときは，一度**Aレジスタに値を移し**，Aレジスタで加算を行った結果（このとき，和はまだAレジスタに入っている），再びもとのレジスタに答えを戻してやる，という三段戦法を用います。

これによると，

Lレジスタを＋2する

には，

LD　A，L
ADD　A，02H
LD　L，A

で実現できることになります。

HLの値はいくつ？

Lレジスタの値を＋2するもう一つの方法は，きわめて単純です。それは，

増加命令（INC）

を用いるものです。

INC命令を実行すると，値が**＋1**されますから，

INC　L
INC　L

のように2回用いれば，

Lレジスタ←Lレジスタ＋2

が実現されます。

ここで突然，①に戻ります。①は，

「スタック領域に0081Hをセットする」

でした。この意味，ちょっと飲み込みにくいかもしれません。

方法は，簡単で

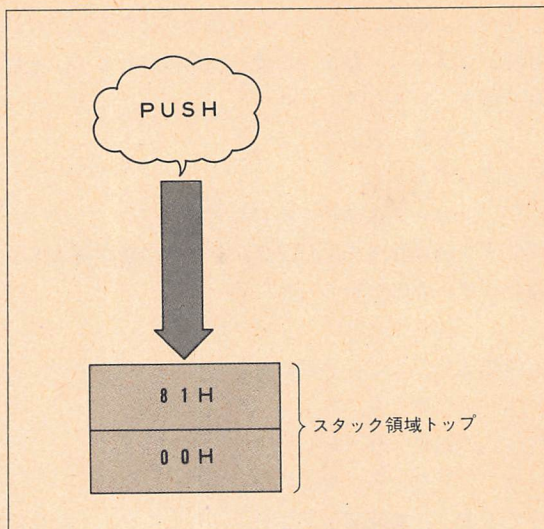
LD　HL，0081H

のように何か適当なレジスタ・ペアに0081Hをセットします。しかる後に，

PUSH　HL

とやります。これで〝スタック領域に0081Hがセット〟されました（**第67図**）。

なんだか何をやっているのか良くわからないかもしれません。マア，あまり気にしないでください。**わざと煙にまいている**のですから。とにかく以上で，大体準備が終りました。

①～⑨までいろいろな操作をしてきて，いまは，HLレジスタの値がある値になっています（いくつだかわかりますか？）。そして，最後の⑩でその

第67図　PUSH命令を使って

HLの指す番地にジャンプ

させる，というのがこの〈チャレンジ〉の主旨です。⑩を実現するには，ピッタリの命令がありまして，

JP　(HL)

がそれです。この命令を行えば，HLの示す番地にジャンプしてくれます。

これで一通り〈チャレンジ〉について，プログラムができあがりました。これをアセンブル前のリストで示したのが，**第68図**。また，そのリストをアセンブルしたのが，**第69図**です。

この第69図のマシン語を入力し，

GC100↘

を実行すれば，C112Hで

HLの指す番地にジャンプ

し，何かが起こるはずです。ところで

HL＝？

HLの値は，いくつになっているでしょうね？

```
;====================
;  KEY LIST
;    (81年 10月 23日)
;====================
;
ORG 0C100H
;
NBAS:EQU 81H
;
MAIN:LD HL,NBAS
PUSH HL
LD A,(DATA)
LD B,A
LD C,10
PUSH BC
POP DE
LD A,D
LD D,E
LD E,A
EX DE,HL
INC HL
INC HL
JP (HL)
;
DATA:DB 70H
;
END
```

第68図　アセンブル前

A72番地へジャンプ

それでは，ＨＬの値に注目して，もう一度

①～⑨

を振り返ってみましょう。

まず①は，あまり関係ありません。スタック領域に値がセットされるだけで，レジスタの値にはあまり関係がないからです。

次の②，③，④で

Ｂレジスタ＝７０Ｈ

になります。また，⑤で

Ｃレジスタ＝０ＡＨ

にセットされますから，ここまでで

ＢＣレジスタ＝７００ＡＨ

になっています。

⑥で，**ＢＣの値をＤＥに移します**から，

ＤＥレジスタ＝７００ＡＨ

になります。もちろん

ＢＣレジスタ＝７００ＡＨ

は変化しません。

次に⑦でＤＥレジスタの**上位バイトと下位バイトが交換**されますから，

ＤＥレジスタ＝０Ａ７０Ｈ

となります。さらに⑧で**この値がＨＬに移され**

ＨＬレジスタ＝０Ａ７０Ｈ

となります。そして最後に⑨で，**Ｌレジスタの値が，＋２されます**から，

ＨＬレジスタ＝０Ａ７２Ｈ

となります。結局，

Ａ７２番地←──ＨＬ

にジャンプすることになりました。本当でしょうか？

```
                     ;====================
                     ;  KEY LIST
                     ;    (81年 10月 23日)
                     ;====================
                     ;
                             ORG   0C100H
                     ;
0081                 NBAS:   EQU   81H
                     ;
C100 218100          MAIN:   LD    HL,NBAS
C103 E5                      PUSH  HL
C104 3A13C1                  LD    A,(DATA)
C107 47                      LD    B,A
C108 0E0A                    LD    C,10
C10A C5                      PUSH  BC
C10B D1                      POP   DE
C10C 7A                      LD    A,D
C10D 53                      LD    D,E
C10E 5F                      LD    E,A
C10F EB                      EX    DE,HL
C110 23                      INC   HL
C111 23                      INC   HL
C112 E9                      JP    (HL)
                     ;
C113 70              DATA:   DB    70H
                     ;
                             END
```


第69図　アセンブル後

```
*SC100
C100 C2-21 01-81 0D-00 25-E5 CD-3A 49-13 C0-C1 38-47
C108 06-0E 3A-0A A6-C5 C2-D1 C6-7A 01-53 3F-5F F5-EB
C110 21-23 FE-23 FF-E9 22-70 52-
*■
```

第70図　Ｓコマンドで入力

```
*DC100,C113
C100 21 81 00 E5 3A 13 C1 47 0E 0A C5 D1 7A 53 5F EB
C110 23 23 E9 70
*■
```

第71図　Ｄコマンドで確認

```
*DC100,C113
C100 21 81 00 E5 3A 13 C1 47 0E 0A C5 D1 7A 53 5F EB
C110 23 23 E9 70
*GC100■
```


第72図　あとはＲＥＴキーを押すだけ

確認するのは，簡単です。**Ｓコマンド**で第69図のマシン語を入力し（**第70図**），**Ｄコマンド**で入力を確認し（**第71図**）

ＧＣ１００（まだ↘は押さない！）

とキーインし（**第72図**），この状態でＲＥＴキーを押せば良いのです。

──しかし！

しかし，待ってくださいよ。

暴走を恐れて

予想では，

Ａ７２番地

にジャンプするはずです。これは，

ＲＯＭ内のルーチン

ですね？　実は，そこには何かの処理を行うルーチンが書かれています。そこで何が行われているかを，これから実験で確かめてみようというわけです。

ところが，**計算**によると，

Ａ７２番地にジャンプ

するはずなのですが，なにせそこにたどりつくまで，（わざと）ゴチャゴチャややっこしいことをやってきました。果して，ここで**ＲＥＴキー**を押してうまく**目的のアドレス**にジャンプできると思いますか？

この程度のプログラムでしたら，**実際に走らせ**，もし失敗して**暴走**を起こし，**プログラムが破壊**されてもたいした被害にはなりません。また最初からキーインすれば良いのですから。

しかし，長いプログラムの場合は，そうはいきません。もし気軽に走らせ，暴走でもしたら目も当てられません。もちろん，念のために**テープにSAVE**してから走らせれば良いのですが，いちいちそんなことをやっているわけにもいきませんね？ DISKをお持ちなら話しは別ですけど。

要は第69図のリストにおいて，Ｃ１１２番地で

ＨＬ＝０Ａ７２Ｈ

の確認ができれば良いわけです。

――それならできる？

そうですね。我々は，「ＰＣ-8001　マシン語入門」（第一巻）のＰ.121で

〝レジスタ表示〞プログラム

を作りました。それを使えば良いのです。

しかし，――。

確かに第一巻で作った〝レジスタ表示〞プログラムを使えば良いのですが，我々はいまや

マシン語の中級者(？)

です。同じ〝レジスタ表示〞プログラムでも，もう少しましなものを使うべきです。

〝ミニ・レジスタ表示〞の準備

たとえば，第一巻Ｐ.121の〝レジスタ表示〞プログラムでは，

プログラム・カウンタ

スタック・ポインタ

の値はわかりません。マシン語の中級者であれば，これらの値も知りたいはずです。

実は，第一巻にもう一つ**〝レジスタ表示〞プログラム**が紹介されていたのを，御存知でしたでしょうか？付章「マシン語入門セミナ開催記」の中で紹介されていたのです。それが，第１図，または第63図のプログラムだったのです。

それでは，いよいよここでそのプログラムに挑戦しますよ。

お待たせ致しました。第63図のプログラムを入力してください。ちょっと長いですが，**101バイト**しかありません。頑張ってキーインしてください。**第73図**が，

ＤＤ０００，Ｄ０６４↘

で確認したところです。

ここまでＯＫでしたら，ひとまず

ＷＤ０００，Ｄ０６４↘

で**カセットにSAVE**しておいてください。

ＬＶ↘

で**録音のチェック**をするのを，お忘れなく。

さて，**〝ミニ・レジスタ表示〞プログラム**（第63図のプログラムのことです）を使うには，**準備**が必要です。それは，ＰＣ-8001 のシステム・ワーク・エリアである

Ｆ１Ｅ３Ｈ～Ｆ１Ｅ５Ｈ

の**３バイト**を書き換える必要があるからです。

この３バイトは，電源ＯＮの状態で，

Ｆ１Ｅ３Ｈ＝Ｃ９Ｈ

```
mon
*DD000,D064
D000 FD E5 DD E5 E5 D5 C5 F5 CD CA 5F 06 25 21 00 00
D010 39 2B 2B F9 E1 11 35 00 19 7E CD 57 02 23 10 F9
D020 CD CA 5F 06 06 E1 CD C0 5E CD D4 5F 10 F7 E1 2B
D030 CD C0 5E CD D4 5F 21 00 00 39 CD C0 5E C3 66 5C
D040 41 46 20 20 20 42 43 20 20 20 44 45 20 20 20 48
D050 4C 20 20 20 49 58 20 20 20 49 59 20 20 20 50 43
D060 20 20 20 53 50
*
```

第73図　１０１バイト分を確認

```
¥SF1E3
F1E3 C9-C3 00-    00-D0 C9-
¥
```

第74図　ワーク・エリアを書き換える

F１E４H＝００H

F１E５H＝００H

にセットされています。それをSコマンドを用いて

F１E３H＝C３H

F１E４H＝００H

F１E５H＝D０H

のように書き換えます（**第74図**）。あなたも実行してみてください。念のために

DF１E３，F１E５↘

で，ちゃんと３バイトが書き換えられたか確めておきましょう（**第75図**）。

```
¥DF1E3,F1E5
F1E3 C3 00 D0
¥
```

第75図　そして確認

実験のねらい

以上で〝ミニ・レジスタ表示〟プログラムを使う準備ができました。まず最初に何か適当なプログラムで実験してみましょう。

そこで取り上げるのが，第１ブロックでやりました第８図の

♥を表示

```
¥DC100,C107
C100 3E E9 CD 57 02 C3 66 5C
¥
```

第76図　８バイトの確認

するプログラムです。これなら短い（８バイト）ですから，すぐに入力できるでしょう。

ということでその**８バイト**を入力し，

DC１００，C１０７↘

で確認したのが，**第76図**です。ここで，このプログラムを走らせることは，すでに第11図でやりました。

♥

が一つ表示されるだけです。

そこで**第77図**を御覧ください。これは，第８図のプログラムを再掲したものです。C１０２Hを見てください。

CALL　OCRT

とあります。これは，**ROM内**にある

１文字出力ルーチン

をCALLしているところです。ここで実験してみようとすることは，このサブルーチンをCALLした後に，

レジスタの値が変化するのか？

ということです。たとえば，このサブルーチンをCALLする前，C１００Hで

LD　A，０E９H

が行われていますから，

Aレジスタの値＝E９H

```
                    ;=================
                    ;  PRINT ♥
                    ;    (CALL 257H)
                    ;=================
                    ;
                            ORG   0C100H
                    ;
0257                OCRT:   EQU   257H          ;OUT CRT (A=CHARACTER CODE)
5C66                GMON:   EQU   5C66H         ;GOTO MONITOR
                    ;
C100 3EE9                   LD    A,0E9H        ;LET A='♥'
C102 CD5702                 CALL  OCRT          ;PRINT '♥'
C105 C3665C                 JP    GMON          ;GOTO MONITOR
                    ;
                            END
```


第77図　８バイトのプログラム

になっています。それが，このサブルーチンをCALLした後でも，まだ

　　Ａレジスタ＝Ｅ９Ｈ

が**保存されているのか**見てみよう，というわけです。

そこで

　　〝ミニ・レジスタ表示〟プログラム

が登場します。すでに

　　Ｆ１Ｅ３Ｈ～Ｆ１Ｅ５Ｈ

の値も書き換え，出番は今か，と待っているところでしたね？

レジスタ類の表示

〝ミニ・レジスタ表示〟プログラムの使い方は，次のとおりです。

〈ミニ・レジスタ表示プログラム〉

① **使用前の準備**

　　Ｆ１Ｅ３Ｈ～Ｆ１Ｅ５Ｈ

の３バイトを書き換える（すでに実行済）

② **使用法**

プログラムの実行をSTOPし，**レジスタの値を見たい番地に**

　　ＦＦＨ

を書き込む。

第77図において，我々が見たいのは，

　　ＣＡＬＬ　ＯＣＲＴ

の後，Ｃ１０５番地でプログラムの実行をSTOPし，レジスタの値を見たいわけです。そこで，Ｓコマンドを使って

　　Ｃ１０５Ｈ＝ＦＦＨ

にセットします（**第78図**）。

```
*SC105
C105 C3-FF 66-
*
```

第78図　FFHを書き込む

その結果，**ダンプ・リスト**（Ｄコマンドで見たリスト）は，**第79図**のように変化します。

第79図　ＦＦＨが書き込まれた様子

これで，いつでもレジスタの値を確められます。

　　ＧＣ１００↘

で♥を表示するプログラムを走らせてみましょう。**第80図**のように♥が表示された後，

　　レジスタ……ＡＦ，ＢＣ，ＤＥ，ＨＬ
　　　　　　　ＩＸ，ＩＹ

　　プログラム・カウンタ……ＰＣ

　　スタック・ポインタ……ＳＰ

の値が表示されました。これを見ると，

　　Ａレジスタ＝Ｅ９Ｈ

で変化していないことがわかります。また，

　　ＰＣ＝Ｃ１０５Ｈ

となっています。これは，

　　プログラムがＣ１０５ＨでＳＴＯＰ

したことを示しています。この番地は，先にプログラムをSTOPさせるために

　　ＦＦＨ

を書き込んだ番地だったですね？

他のレジスタについても，御覧のように調べることができます。おそらく，ＳＰ（スタック・ポインタ）の見方がわかりずらいかと思いますが，これについてもおいおい説明していくことに致します。

なお，〝ミニ・レジスタ表示〟プログラムが働いた後でも，オリジナルのプログラムは，消えずに残っています（**第81図**）。

第80図　レジスタ類が現われる

第81図　もとのプログラムは保存されている

第8章

リロケータブルの世界

```
DA80: 28 05 15 20 F8 18 DE 3A 67 EA CB AF C9 E5 D5 CD ; 08A5
DA90: 99 DA D1 0D 20 F8 E1 23 C9 EB 50 3A 67 EA CB EF ; 09B6
DAA0: D3 40 2B 7C A7 28 15 15 20 FB 50 3A 67 EA CB AF ; 0720
DAB0: D3 40 2B 7C A7 28 05 15 20 FB 18 DE C9 3E 98 2E ; 067E
DAC0: 01 CD 60 D9 3E 38 2E 02 C3 60 D9 2A CD E3 7C B5 ; 07B4
DAD0: 20 1E CD 95 D9 E6 7F C0 CD 95 D9 E6 01 20 03 67 ; 084A
DAE0: 18 04 3E FF 26 47 32 CF E3 2E 01 22 CD E3 18 2F ; 05F2
DAF0: CD 2C DB 21 CE E3 3A CF E3 A7 20 0F 7E FE 47 20 ; 084B
DB00: 07 21 00 00 22 CD E3 C9 34 18 0C 7E A7 20 07 21 ; 0488
DB10: 00 00 22 CD E3 C9 35 01 06 20 CD 0D DA 18 00 2A ; 04ED
DB20: CD E3 CD 0C D9 11 29 E1 1A C3 1D D9 01 02 09 AF ; 070B
DB30: 2A CD E3 C3 4E D9 DB 00 FE EF CC 98 D4 FE BF CC ; 0B4D
DB40: A6 D4 CD 49 DB DC 60 DB C9 DB 09 FE BF 37 3F 28 ; 098A
DB50: 05 AF 32 AF E3 C9 21 AF E3 7E A7 C0 2F 77 37 C9 ; 087F
DB60: 2A AD E3 7C B5 C0 CD 75 DB 3A AC E3 3C 3C 67 2E ; 089E
DB70: 17 22 AD E3 C9 01 1A 30 C3 0D DA 2A AD E3 7C B5 ; 0772
DB80: C8 E5 CD A7 DB E1 2D CD B1 DB 22 AD E3 CD 0C D9 ; 0AC7
DB90: 7E A7 20 03 36 0F C9 E5 F5 CD BC DB F1 E1 CD 34 ; 0967
DBA0: DC CD 88 DC C3 E2 D6 CD 0C D9 7E FE 0F C0 AF 77 ; 0AAB
DBB0: C9 7D A7 C0 21 00 00 22 AD E3 E1 C9 3A AD E3 FE ; 08F2
DBC0: 03 D0 CD EB DB 3E E8 2E 02 CD 60 D9 CD 00 DC CD ; 0938
DBD0: 85 D9 CD 79 D9 CD BD DA CD 2C DB CD 7C D8 21 00 ; 09F7
DBE0: 00 22 CD E3 22 AD E3 E1 E1 E1 C9 2A AD E3 2D 25 ; 08FC
DBF0: 25 E5 CD 20 D7 E1 AF 01 02 05 CD 4E D9 C3 2C DB ; 0824
```

ねがいましては　りろけーたぶる………

マシン語の移動

もうしばらく第77図のプログラムで実験を続けましょう。

いまあなたのＰＣのメモリには，第81図の状態でプログラムが入っていると思います。Ｃ１０５Ｈが，ＦＦＨに書き換えられていますから，**Ｓコマンド**でなおしましょう（**第82図**）。

　　ＤＣ１００，Ｃ１０７↘

を実行すると(**第83図**)，元の第77図のプログラムに戻っているのが確認できます。もちろん

　　ＧＣ１００↘

で♥が表示されます（**第84図**）。

第82図　Ｓコマンドでなおす

第83図　もとに戻った

いまこの第83図の状態のマシン語を，そのまま

　　ソックリＤ０００Ｈに移す

と，どうなるでしょうか？

あなたも**第85図**のように，**Ｓコマンド**を使ってマシン語を

　　Ｄ０００Ｈ～Ｄ００７Ｈ

に移してみてください。

　　ＤＤ０００，Ｄ００７↘

で確認します（**第86図**）。

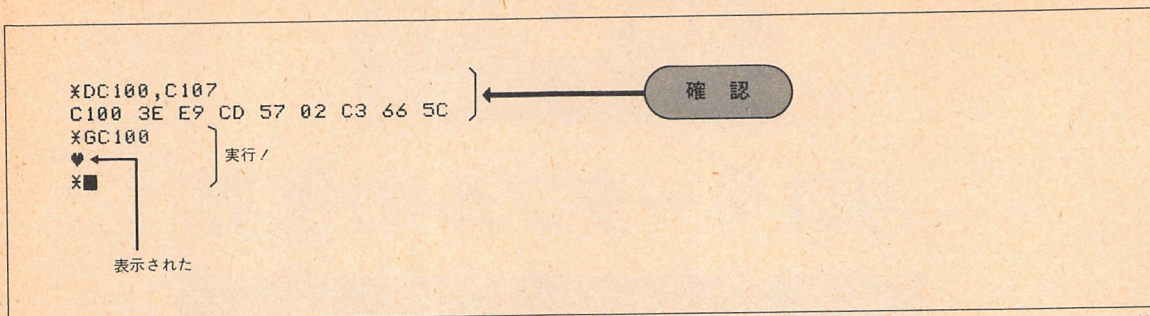
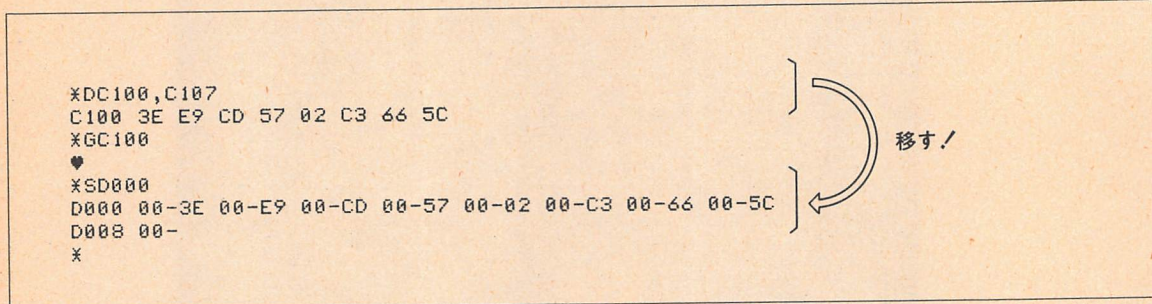
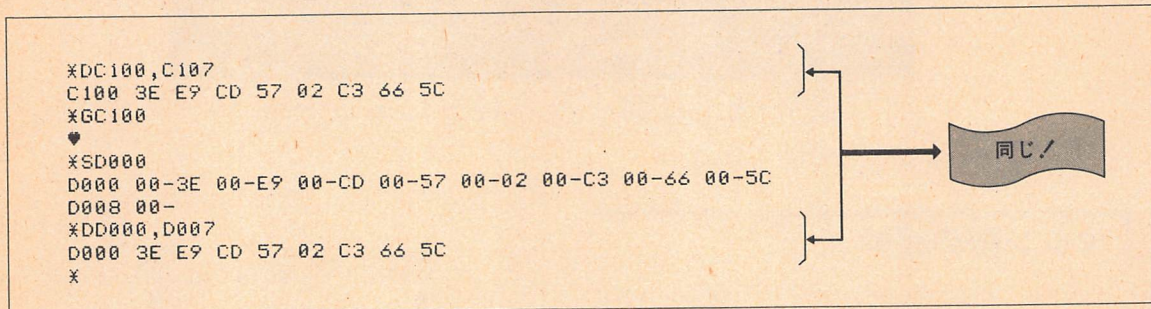
ここで，もし

　　ＧＤ０００↘

を実行すると，どうなると思いますか？　おそらく，

第84図　実行も可能

```
¥DC100,C107
C100 3E E9 CD 57 02 C3 66 5C
¥GC100
♥
¥SD000
D000 00-3E 00-E9 00-CD 00-57 00-02 00-C3 00-66 00-5C
D008 00-
¥
```

移す！

第85図　D000H～D007Hにマシン語を移す

第86図　Dコマンドで確認

二つの意見に分かれるのではないでしょうか。

〈意見その1〉

Ｃ１００番地から組んだプログラムを，そのまま他の番地に移しても動くはずがない。だから，

ＧＤ０００↘

を実行すると，何が起こるかわからない。

〈意見その2〉

マシン語をどの番地から入力したって，もともとは同じプログラムなのだから，

ＧＤ０００↘

を実行すれば，**同じように動くはず**だ。

あなたは，どちらの意見を採用しますか？　または，これらとは異なる意見をお持ちですか？

軍配一夢は広がる

それでは結論を得るため，プログラムを走らせてみましょう。

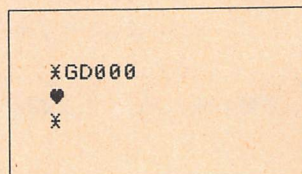
ＧＤ０００↘

結果は，**第87図**のとおりです。問題なく♥が表示されました。どうやら〈意見その2〉さんに軍配があがったようです。

```
¥GD000
♥
¥
```

第87図　プログラムは動いた

――ということは。

もし〈意見その2〉さんが正しいとすれば，これは大変なことになります。ＰＣ-8001，否，ひいては80系ＣＰＵのユーザー全員に朗報をもたらすことになるのです。

たとえば，次のようなことが可能になります。

① 8000H～BFFFHで開発されたマシン語のプログラムでも，C000H以後の適当なところに入力することで，今まで**32Kシステムでなければ動かなかったプログラムが，16Kシステムでも動く**ようになる。

② どんな番地のプログラムでも自分の好きな番地に移せるから，**同時に何本ものプログラムをメモリに**入れておき，次々と違うプログラムを走らせることができるようになる。

他にもアイデア次第でいろいろなことが可能になります。これ，凄いと思いませんか？

――ただし，これらのことは〈意見その2〉さんの意見が正しかった時のことです。

説がくずれる

まだ判定をくだすのは，早計です。ここで，もう一つ実験をしてみましょう。

ここで取り上げるのは，第1ブロック，第26図で取り上げた〝**文字列出力ルーチン**〟に関するプログラムです。**第88図**として再掲してあります。このプログラムを入力し，Dコマンドでダンプ・リストを取り，Gコマンドで実行したのが，**第89図**に掲げてあります。

＊＊ PC―8001 マイコン ＊＊

と表示されるのでしたね。

続いてこの**プログラムを消去**します。一度電源を消すか，DISKを接続してある方は電源を落とすのは面倒でしょうから，**第90図**のように**Sコマンド**を使うと良いでしょう。**0を押しっ放し**にしておくと，オート・

```
                        ;================
                        ;  CALL 52EDH
                        ;     (82.10.6)
                        ;================
                        ;
                                ORG   0C100H
                        ;
52ED                    MSG:    EQU   52EDH          ;HE=POINTER,END MARK=0
5C66                    MON:    EQU   5C66H          ;GOTO MONITOR
                        ;
C100 2109C1                     LD    HL,DATA
C103 CDED52                     CALL  MSG
C106 C3665C                     JP    MON
                        ;
C109 2A2A2050           DATA:   DC    '** PC-8001 ﾏｲｺﾝ **'  ;ﾒｯｾｰｼﾞ DATA
C10D 432D3830
C111 303120CF
C115 B2BADD20
C119 2A2A
C11B 00                         DB    0
                        ;
                                END
```

第88図　もう一つの実験用プログラム

```
*DC100,C11B
C100 21 09 C1 CD ED 52 C3 66 5C 2A 2A 20 50 43 2D 38
C110 30 30 31 20 CF B2 BA DD 20 2A 2A 00
*GC100
** PC-8001 ﾏｲｺﾝ **
*
```


第89図　ダンプ・リスト＆実行例

```
¥SC100
C100 21-00 09-00 C1-00 CD-00 ED-00 52-00 C3-00 66-00
C108 5C-00 2A-00 2A-00 20-00 50-00 43-00 2D-00 38-00
C110 30-00 30-00 31-00 20-00 CF-00 B2-00 BA-00 DD-00
C118 20-00 2A-00 2A-00 00-00 98-
¥DC100,C11B
C100 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
C110 00 00 00 00 00 00 00 00 00 00 00 00
¥
```

←Sコマンドで消去

←消えた、消えた！

第90図　プログラムの消去と確認

リピートがきいて，簡単にプログラムを消去することができます。

次に第88図のプログラムを，他の番地（C100H以外ならどこでも結構です）に入力します。ここでは，

　　E000H～E01BH

に入力することにしました。**第91図**のようにSコマンドを使って，せっせと入力します。そして，

　　DE000，E01B↘

で入力の確認です。

```
¥SE000
E000 2A-21 A0-09 EF-C1 CD-CD 95-ED 40-52 38-C3 28-66
E008 59-5C 16-2A 00-2A 2A-20 50-50 EB-43 19-2D EB-38
E010 2A-30 79-30 EF-31 CD-20 95-CF 40-B2 DA-BA 68-DD
E018 E0-20 F1-2A 79-2A CD-00 03-
¥DE000,E01B
E000 21 09 C1 CD ED 52 C3 66 5C 2A 2A 20 50 43 2D 38
E010 30 30 31 20 CF B2 BA DD 20 2A 2A 00
¥
```

第91図　E000H～E01BHに移す

さあ，これでこのプログラムを走らせると，**〈意見その2〉さんの説**によれば，今回も第89図のように

　　＊＊　PC―8001　マイコン　＊＊

と表示されるはずです。

　　GE000↘

で走らせてみましょう。結果は**第92図**のとおりです。今度は，何も表示されませんでした。

第92図　実行！

さあ，すると理由がわからなくなってきました。の実験においては，〈意見その2〉さんのいう

　　開発されたマシン語は，メモリ上のどの領域に置いても良い

とする意見が正しいように見えました。しかし，今度の実験では，どうやらその主張はくずれたかのように見えます。いったい，どちらの意見が正しいのでしょうか？

ハンド逆アセンブル

それでは，これから

　　正しい結論を得る

ために，

　　ハンド逆アセンブル

という手法を使ってみます。

第86図で入力した

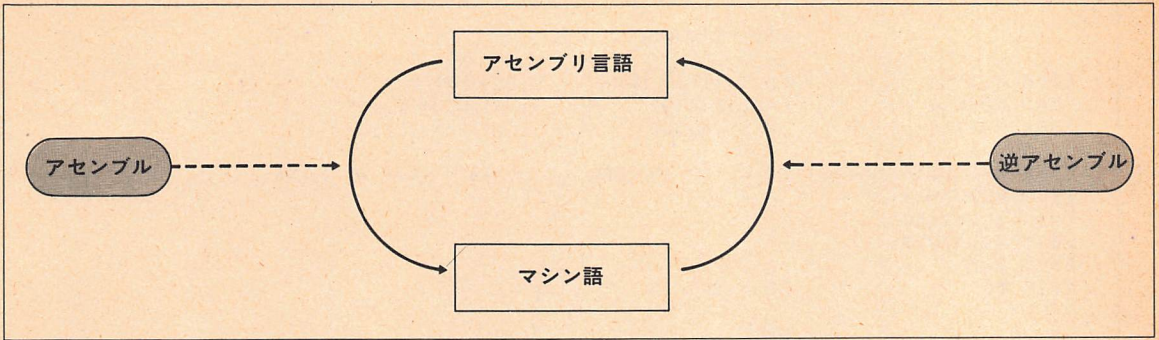
　　D000H～D007H

の

　　8バイトのデータ
　　└（(注)これがマシン語である，とはまだ断定できない。）

は，もともとは

　　C100H～C107H

の上で開発された**マシン語**です。それを単純にそのま

第93図　逆アセンブル

ま他の領域に移したとしても，それが

　　　正しいマシン語のプログラムになっている

とは，**断定できないわけです。**

　それを確認するには，その

　　　8バイトのデータ

が，**正しいプログラムであるかどうか**を調べてみなければなりません。そのためには，付録の

　　　〝機械語↔ニーモニック対応表〟

を使用します。

　第86図においては，データが，

　　　D000H＝3EH
　　　D001H＝E9H
　　　D002H＝CDH
　　　　⋮
　　　D007H＝5CH

のようにメモリに格納されています。いま我々が行おうとしていることは，これらのデータを

　　　マシン語である

と見なし，

　　　アセンブリ言語に変換

してみよう，ということです。いわば，**アセンブル作業の逆**のことを行うわけです。普通，この作業を

　　　逆アセンブル

といっています（**第93図**）。

逆アセンブル・リストの解析

　それでは，具体的に**ハンド逆アセンブル**をしていきます。まず最初のデータである3EHを付録の表で調べます。

　　　3EH＝LD　A，　n

となっています。**nは，1バイトのデータ**ですから，次のE9Hと一緒にして，

　　　3E　E9　→　LD　A，3EH

と逆アセンブルされます。

　続いて次のCDHを調べますと，

　　　CDH＝CALL　nn

になっています。**nnは，2バイトのデータ**ですから，次の57H，02Hと一緒にして

　　　CD　57　02　→　CALL　257H

と逆アセンブルされます。

　以下，この要領で逆アセンブルしていくと，

　　　LD　　A，3EH
　　　CALL　257H
　　　JP　　5C66H

と逆アセンブルされるのがおわかりでしょう。これをちゃんと逆アセンブル・リストの形で示したのが，**第94図**です。

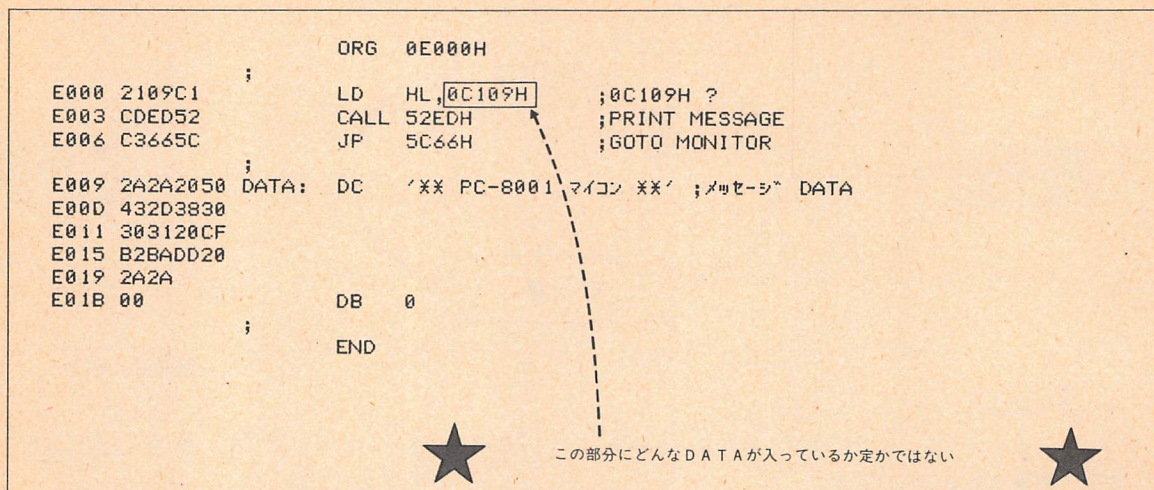
　次にこの逆アセンブルされた**リストを解読**してみます。

① Aレジスタに3EH（♥のキャラクタ・コード）をセットする。

```
                        ORG   0D000H
                 ;
D000 3EE9               LD    A,0E9H          ;LET A='♥'
D002 CD5702             CALL  257H            ;PRINT '♥'
D005 C3665C             JP    5C66H           ;GOTO MONITOR
                 ;
                        END
```

このプログラムは、意味のあるものとなっている

第94図　ハンド・逆アセンブルの完成

```
                        ORG   0E000H
                   ;
E000 2109C1             LD    HL,0C109H        ;0C109H ?
E003 CDED52             CALL  52EDH            ;PRINT MESSAGE
E006 C3665C             JP    5C66H            ;GOTO MONITOR
                   ;
E009 2A2A2050      DATA: DC   '** PC-8001 ﾏｲｺﾝ **' ;ﾒｯｾｰｼﾞ DATA
E00D 432D3830
E011 303120CF
E015 B2BADD20
E019 2A2A
E01B 00                 DB    0
                   ;
                        END
```


第95図　意味のないリストのできあがり

② ＲＯＭ内の〝１文字出力ルーチン〞をCALLする。

③ モニタに戻る。

以上のように，ちゃんと理にかなったプログラムになっているのがわかります。これを走らせれば，♥が表示されるのは，当り前ですね。第77図のプログラムは，どこの番地に移しても，元のプログラムに逆アセンブルされます。したがいまして，**どの番地にでも移すことが可能**です。

データ領域も移動する

次に第88図のプログラムを

Ｅ０００Ｈ～Ｅ０１ＢＨ

に移した第91図のデータを，ハンド逆アセンブルしてみましょう。

Ｅ０００Ｈ＝２１Ｈ

Ｅ００１Ｈ＝０９Ｈ

⋮

Ｅ０１ＢＨ＝００Ｈ

ですから，

ＬＤ　　ＨＬ，Ｃ１０９Ｈ

ＣＡＬＬ　５２ＥＤＨ

ＪＰ　　５Ｃ６６Ｈ

と**逆アセンブル**できます。ここで注意が必要なことは，

Ｅ００９Ｈ～Ｅ０１ＢＨ

のデータについては，

＊＊　ＰＣ－８００１　マイコン　＊＊

の文字列の部分に当たりますから，

逆アセンブルしてはいけない！

ということです。したがってこの部分は，

ＤＣ　または　ＤＢ

を使って**データ**として残しておきましょう。

でき上がった逆アセンブル・リストが，**第95図**です。このリストを解析していきましょう。

① ＨＬレジスタに，文字列先頭の番地Ｃ１０９Ｈをセットする。

② その文字列を表示する。

③ モニタに戻る。

分かりましたか？　もうピンときたでしょう。この**プログラムが正しく動かない理由が？**

このプログラムは，

文字列出力ルーチン

を用いるものです。それは，

ＨＬ＝文字列の先頭アドレス

にセットしてからＣＡＬＬするものでした。第95図のプログラムも，Ｅ０００Ｈで

ＬＤ　　ＨＬ，０Ｃ１０９Ｈ

と，**文字列の先頭番地をセット**しています。しかし，このプログラムでは，

文字列の先頭＝Ｅ００９Ｈ

であって，Ｃ１０９Ｈではありません。

第88図のように，このプログラムはもともと

Ｃ１００Ｈ～Ｃ１１ＢＨ

のもとで開発されたものであり，その時は

文字列の先頭＝C109H

で良かったのです。そのマシン語をそのまま

E000H～E01BH

に移したため，

文字列の先頭＝E009H

に移動したにもかかわらず，HLレジスタにセットした値は元のままだったため，何も表示されなかったのです。これが，このプログラムがうまく動かなかった理由です。

珍説の崩壊

したがいまして，第91図のプログラムをうまく動かそうとするなら，

LD HL,0C109H――(誤)

LD HL,0E009H――(正)

のように訂正してやれば良いのです。すなわち

E002H＝C1H――――(誤)

↓

E002H＝E0H――――(正)

のように変えます。これは，**Sコマンド**で簡単にできます(**第96図**)。**第97図**が，訂正後のプログラムです。これを

GE000↘

で走らせれば，もちろん正しく

＊＊ PC―8001 マイコン ＊＊

と表示されます(**第98図**)。

以上，我々が実験してきたことをまとめますと，次のようになります。

> **〈教 訓〉**
> Z―80のマシン語は，そのまま他の領域に転送しても，**正しく動くとは限らない。**

残念ながら，〈意見その2〉さんの意見は，もろくも崩れてしまいました。

リロケータブル

これで，マシン語のプログラムには，

二種類のタイプ

があることが，おわかりになったと思います。一つは，

メモリ上のどの部分にでも移せるもの

であり，

そのままではメモリ上の他の部分には
移せない

ものです。前者のように，メモリ上の空いているところなら，そのまま移すことが可能なプログラムのことを，

第96図 Sコマンドで修正

```
*DE000,E01B
E000 21 09 E0 CD ED 52 C3 66 5C 2A 2A 20 50 43 2D 38
E010 30 30 31 20 CF B2 BA DD 20 2A 2A 00
*■
```

第97図 修正OK！

第98図 今度はOK！

```
                  ;===================================
                  ;  MINI REGISTER DISPLAY
                  ;    (81年 3月 18日):BY K.TSUKAGOSHI
                  ;===================================
                  ;
                         ORG   0D000H
                  ;
0257              OCRT:  EQU   257H            ;OUT CRT
5C66              GMON:  EQU   5C66H           ;GOTO MONITOR
5EC0              PRHL:  EQU   5EC0H           ;PRINT HL
5FCA              CRLF:  EQU   5FCAH
5FD4              PSPC:  EQU   5FD4H           ;PRINT SPACE
                  ;
D000 FDE5         MAIN:  PUSH IY               ;REGISTER STORE FOR DISPLAY
D002 DDE5                PUSH IX
D004 E5                  PUSH HL
D005 D5                  PUSH DE
D006 C5                  PUSH BC
D007 F5                  PUSH AF
                  ;
D008 CDCA5F              CALL CRLF             ;CARRIAGE LINE FEED,PREPRA FOR DISPLAY
                  ;
D00B 0625                LD   B,37             ;'AF-SP'=37 CHARACTER
D00D 210000              LD   HL,0             ;LET HL=SP
D010 39                  ADD  HL,SP
D011 2B                  DEC  HL
D012 2B                  DEC  HL
D013 F9                  LD   SP,HL
D014 E1                  POP  HL
D015 113500              LD   DE,53            ;HL=ADR(DATA)
D018 19                  ADD  HL,DE
                  ;
D019 7E           MA1:   LD   A,(HL)           ;PRINT 'AF-SP'
D01A CD5702              CALL OCRT
D01D 23                  INC  HL
D01E 10F9                DJNZ MA1
                  ;
D020 CDCA5F              CALL CRLF
                  ;
D023 0606                LD   B,6              ;6 REGISTERS
D025 E1           MA2:   POP  HL
D026 CDC05E              CALL PRHL
D029 CDD45F              CALL PSPC
D02C 10F7                DJNZ MA2
                  ;
D02E E1                  POP  HL               ;PRINT PC
D02F 2B                  DEC  HL
D030 CDC05E              CALL PRHL
D033 CDD45F              CALL PSPC
                  ;
D036 210000              LD   HL,0             ;PRINT SP
D039 39                  ADD  HL,SP
D03A CDC05E              CALL PRHL
                  ;
D03D C3665C              JP   GMON             ;GOTO MONITOR
                  ;
D040 41462020     DATA:  DB   'AF    BC    DE    HL    '
D044 20424320
D048 20204445
D04C 20202048
D050 4C202020
D054 49582020            DB   'IX    IY    PC    SP'
D058 20495920
D05C 20205043
D060 20202053
D064 50
                  ;
                         END
```

第99図　三度目の登場

リロケータブル (relocatable)
なプログラム

といいます。

> **〈リロケート〉 relocate**
> メモリ上で，プログラムの全部または一部を移動させること。

リロケータブルなプログラムというのは，便利なことは便利なのですが，実際は非常に少ないものです。特に**長いプログラム**は，まず**リロケータブルではない**と考えて良いでしょう。

リロケータブルなプログラムとそうでないプログラムとの見分けは，大体次のようになります。

> **〈リロケータブルでないプログラム〉**
> ① 内部に**データ領域**（たとえば，文字列）を持つもの。
> ② **ＪＰ命令**を使っているもの。ただし，ＪＲ命令の使用はかまわない。
> ③ 内部にサブルーチンを持つもの。

これらの条件のいずれか一つにでも触れれば，そのプログラムは，大体

リロケータブルではない！

といえます。この３条件に触れないプログラムは，まずあり得ませんから，ほとんどのプログラムは，リロケータブルでないと考えて良いでしょう。

二つのプログラムをLOADして

前節で

リロケータブルなプログラム
の見分け方

を取り上げました。そこで登場するのが，**第99図**のプログラムです。お馴染みの

〝ミニ・レジスタ表示〟プログラム

ですね。すでに第１図，第63図でお目にかかりました。これで三度目の登場，**大スター**です。

さて，第99図のリストを良く御覧ください。このプログラムは，

リロケータブル

ですか。

たとえば①**の条件**を見ると，プログラム内部

Ｄ０４０Ｈ～Ｄ０６４Ｈ

に**データ領域**を持っていますから，明らかにこのプログラムは，

リロケータブルではない！

といえそうです。

さて，ここで新しいプログラムが登場します。**第100図**がそれです。何のプログラムかは，まだ秘密にして

```
                    ;====================
                    ;  MOVE PROG
                    ;    (81年 10月 31日)
                    ;====================
                    ;
                            ORG   0C100H
                    ;
5C66                MON:    EQU   5C66H          ;GOTO MONITOR
                    ;
C100 AF             MAIN:   XOR   A              ;CY=0
C101 1100D0                 LD    DE,0D000H      ;MINI START
C104 2164D0                 LD    HL,0D064H      ;MINI END
C107 ED52                   SBC   HL,DE
C109 23                     INC   HL             ;HL=BYTES
C10A E5                     PUSH  HL
C10B C1                     POP   BC             ;BC=BYTES
C10C 2100D0                 LD    HL,0D000H      ;FROM
C10F 1100E0                 LD    DE,0E000H      ;TO
C112 EDB0                   LDIR                 ;MOVE
C114 C3665C                 JP    MON
                    ;
                            END
```


第100図　新規プログラムの登場

おきましょう。

何はともあれ，このプログラムを**Sコマンド**であなたのPCにキーインしてください。続いて，第99図の〝ミニ・レジスタ表示〟プログラムも入力してください。これはすでにカセットに録音してありましたね。

L↘

で簡単にロードできます。

第101図を御覧ください。図のように，今あなたのPCには，

C100H〜C116H
新規登場プログラム（第100図）
D000H〜D064H
ミニ・レジスタ表示（第99図）

の**二本のプログラムが同居**していることになります。二本のプログラムを同時にメモリに常駐させるなんて，**BASIC**ではできませんでしたね？　ここが，マシン語の便利なところです。

（注）BASICでも，マシン語でちょっとイタズラをすると，同時に**複数のプログラム**をメモリに同居させることができます。

ブロック転送

続いて何をするか？　ハイ，

C100H〜C116H

にある**新規登場プログラム**を走らせます。しかし，その前に後々のことを考えて

DD000，D064↘

で〝ミニ・レジスタ表示〟の**ダンプ・リスト**を表示させておき，さらに一回

↘（RETキー）

を押してください。これは**表示を見やすくする**ためで，

＊

が一つ表示されます。

これで準備ができました。

GC100↘

でプログラムを走らせてみてください。エ？　何も起こらない？　結構。続いて

DE000，E064↘

を実行してください（**第102図**）。そして，良く画面を御覧になってください。

新たに表示された

E000H〜E064H

のデータを良く眺めてください。

「どこかで見たことのあるデータだな？」

と思いませんか？

もうお気付きでしょう。

D000H〜D064Hのデータ
⇅同じ
E000H〜E064Hのデータ

になっていますね？

そうです。

C100H〜C116H

に入っている**新規登場プログラム**は，

D000H〜D064H

に入っている**〝ミニ・レジスタ表示〟プログラム**を，

E000H〜E064H

に転送するプログラムだったのです。このようなプログラムを

ブロック転送プログラム

```
*DC100,C116
C100 AF 11 00 D0 21 64 D0 ED 52 23 E5 C1 21 00 D0 11
C110 00 E0 ED B0 C3 66 5C
*
*DD000,D064
D000 FD E5 DD E5 E5 D5 C5 F5 CD CA 5F 06 25 21 00 00
D010 39 2B 2B F9 E1 11 35 00 19 7E CD 57 02 23 10 F9
D020 CD CA 5F 06 06 E1 CD C0 5E CD D4 5F 10 F7 E1 2B
D030 CD C0 5E CD D4 5F 21 00 00 39 CD C0 5E C3 66 5C
D040 41 46 20 20 20 42 43 20 20 20 44 45 20 20 20 48
D050 4C 20 20 20 49 58 20 20 20 49 59 20 20 20 50 43
D060 20 20 20 53 50
*
```

新規登場プログラム（C100〜C116）
ミニ・レジスタ表示（D000〜D064）

第101図　2つのプログラムをメモリに

```
D000 FD E5 DD E5 E5 D5 C5 F5 CD CA 5F 06 25 21 00 00
D010 39 2B 2B F9 E1 11 35 00 19 7E CD 57 02 23 10 F9
D020 CD CA 5F 06 06 E1 CD C0 5E CD D4 5F 10 F7 E1 2B
D030 CD C0 5E CD D4 5F 21 00 00 39 CD C0 5E C3 66 5C
D040 41 46 20 20 20 42 43 20 20 20 44 45 20 20 20 48
D050 4C 20 20 20 49 58 20 20 20 49 59 20 20 20 50 43
D060 20 20 20 53 50
¥
¥GC100
  ○
¥DE000,E064
E000 FD E5 DD E5 E5 D5 C5 F5 CD CA 5F 06 25 21 00 00
E010 39 2B 2B F9 E1 11 35 00 19 7E CD 57 02 23 10 F9
E020 CD CA 5F 06 06 E1 CD C0 5E CD D4 5F 10 F7 E1 2B
E030 CD C0 5E CD D4 5F 21 00 00 39 CD C0 5E C3 66 5C
E040 41 46 20 20 20 42 43 20 20 20 44 45 20 20 20 48
E050 4C 20 20 20 49 58 20 20 20 49 59 20 20 20 50 43
E060 20 20 20 53 50
¥
```


第102図　新規登場プログラムを走らせる

と呼んでいます。

> **〈ブロック転送〉**
> 　メモリ上のある領域（ブロック）にあるデータ（プログラムやデータ）を，他の領域に移すこと。

LDDR命令

第102図を御覧になって，あなたは一つの**疑問**に出会ったのではないでしょうか？

「"ミニ・レジスタ表示" プログラムは，内部にデータを持っているから，**リロケータブルなプログラムではない**。にもかかわらず，そのプログラムを他の領域に転送しても**あまり意味がない**のではないか？」

――ごもっともです。

この疑問については，あとでお答えするとして，その前に第100図の

　　　ブロック転送プログラム

の中身を説明しておきましょう。と申しますのは，このプログラムが，**非常に有益なプログラム**だからです。

マシン語をいじくっていますと，

　　　ある領域のデータ
　　　　⇓〈転送〉
　　　他の領域

したいことが良く起こります。そのたびに，わざわざ**キーインしなおす**のは面倒です。そんな時，第100図のプログラムをちょっと応用すれば，プログラムを簡単に

　　　ブロック転送

することができます。

Z―80の命令の中には，実は

　　　ブロック転送を行ってくれる命令

があります。それが

　　　ＬＤＩＲ命令

です。第100図のプログラムでは，このLDIRを用いて転送を行っています。

（注）ブロック転送を行う命令は，他にもLDDRがあります。

転送バイト数を求める

LDIR命令の使い方は，次のとおりです。

- **ＢＣレジスタ**＝転送するバイト数
- **ＤＥレジスタ**＝転送後の先頭アドレス
- **ＨＬレジスタ**＝転送前の先頭アドレス

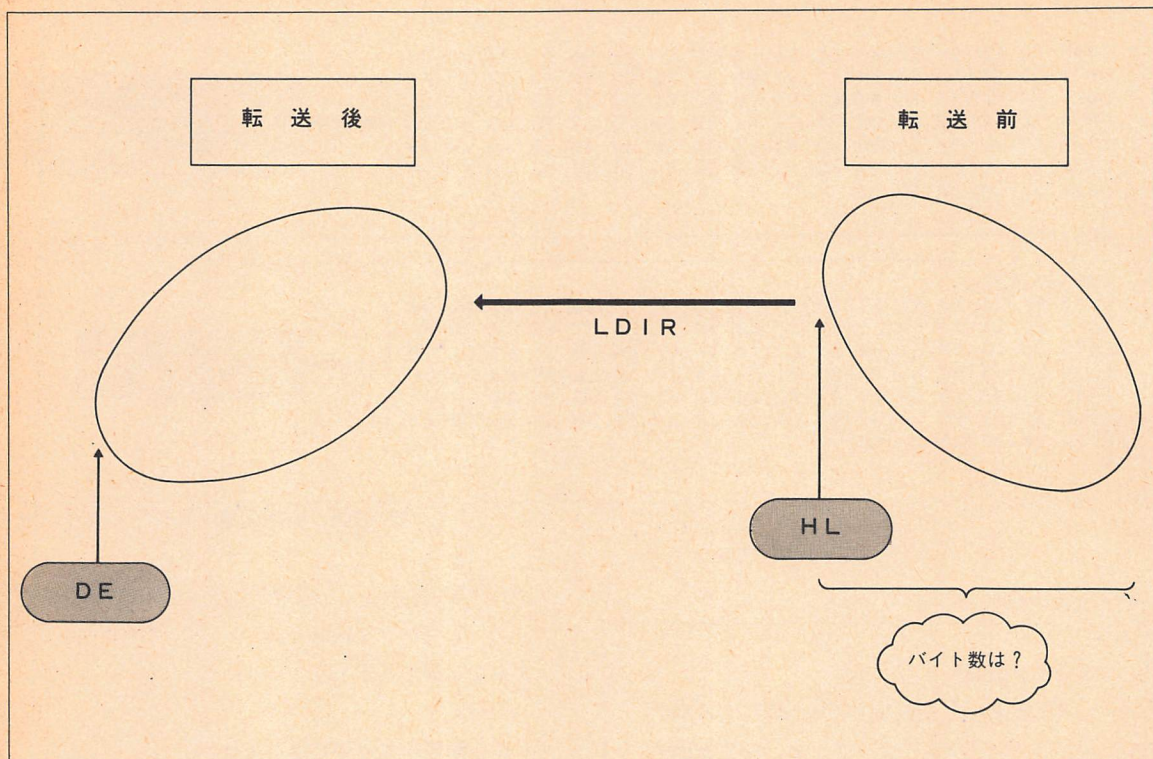
この三つの条件をセットした後に

　　　ＬＤＩＲ

とすれば，ブロック転送されます。第100図の例でしたら，

　　　転送前の先頭アドレス＝Ｄ０００Ｈ
　　　転送後の先頭アドレス＝Ｅ０００Ｈ

ですから，

```
LD  HL, 0D000H
LD  DE, 0E000H
```

第103図　ＬＤＩＲの使用前の準備

とセットします。これは問題ないのですが，

転送するバイト数

は，どのように計算したら良いでしょうか？(**第103図**)。

一般にブロック転送を行う場合，

転送前の先頭アドレス――――①
転送前の最終アドレス――――②
転送後の先頭アドレス――――③

の三条件なら簡単にわかるのですが，LDIRで必要な条件（パラメータ）は，

転送前の先頭アドレス――――①
転送するバイト数――――――②′
転送後の先頭アドレス――――③

の三条件となっています。すなわち，

②≠②′

と**若干のズレ**が生じています。そこで

転送すべきバイト数を求める

ため，元のプログラムのリストを見ながら，原始的に指で数えたりします。しかし，これではいかにも**芸が無い**ですね。

そこで，このバイト数を求めるのをＰＣにやらせてしまおう，ということになります。これは，

（転送バイト数）

＝（最終アドレス）－（先頭アドレス）＋１

という関係を利用すれば，比較的簡単にプログラム化することができます。すなわち，先の条件でいえば，

②′＝②－①＋１

という関係式が成立しますから，これを利用するのです。

2バイトの減算命令

ところで，

最終アドレス＝**2バイトの数**

先頭アドレス＝**2バイトの数**

ですから，転送バイト数を求めるには，

2バイトの減算命令

を実施する必要があります。

そこで**Z-80の命令**の中に，

2バイトの減算を行う命令

がないか，付録の

〝Ｚ―80活用表″

を捜してみます。ところが，どうもピッタリの命令がありませんね？　これに最も近い命令は，

SBC　HL，X

（X＝BC，DE，HL，SP，IX，IY）

位なものです。そこで**SBC命令**を用い，

SBC　HL，DE――――①

で**2バイトの減算を行う**ことを考えてみます。

①の命令は，

HL←HL－DE－CY――――②

（CY：キャリー・フラグ）

を行うものです。我々がやりたいのは，

HL＝HL－DE――――③

なのですが，そういう命令はないので②を使うしかありません。②と③を比べると，

－CY

の部分が余計です。CYは，**キャリー・フラグ**のことで，その値は1か0です。そのどちらであるかは，それまでの状況によりますので，定かではありません。もし，

CY＝0

であれば，

HL－DE－CY

＝HL－DE－0

＝HL－DE

で好都合なのですが――。

それには，

SBC　HL，DE

を使う直前に，強制的に

CY＝0

にしてしまえば良い，ということは容易に想像がつきます。かくて我々は，

CY＝0

にする命令を捜すことになります。

CY＝0にする

そこで再び，

〝Z－80活用表〟

をめくり，**該当する命令はないか**とあさってみます。しかし，残念ながらそのような命令は見当りません。さあ，あなたならどうしますか？　本当に考えてみてくださいよ。

ここで二つ程，その**解決法**を御紹介します。

〈解決法その1〉

解決法その1は，なんとか工夫して

CY＝0

にしてしまうというものです。

CY＝0

にする命令はないが，

CY＝1

にする命令（SCF）はある。これに

CCF命令：CYの値を逆転する

を組合わせれば，解決するのではないでしょうか？

以上の結果得られるプログラムは，

```
SCF         ←――――――CY＝1
CCF         ←――――――CY＝0
SBC  HL，DE  ←――2バイト減算
```

です。

〈解決法その2〉

解決法その2は，アッサリと

CY＝0

にするのをあきらめてしまう方法です。かつてZ－80の前身である8080には，

SBC命令

はありませんでした。したがってその時は，2バイトの減算を行うのに，**1バイト減算命令を組み合わせて**行っていました。ここでは詳しくは説明しませんが，方法だけを簡単に御紹介しておきます。たとえば，

HL＝HL－DE

を行いたいのでしたら，

```
LD   A，L
SUB  E
LD   L，A
LD   A，H
SBC  A，D
LD   H，A
```

で出来ます。

秘伝と解析

前節ではややっこしい方法を二つ御紹介しましたが，実は一発で

CY＝0

にする方法があります。それは，論理演算の命令を用いるのです。

```
AND  A
OR   A
XOR  A
```

のどれでも構いません。このうちの一つ，たとえばXORを用いれば，

```
XOR  A
SBC  HL, DE
```

のようにして2バイトの減算命令を実行できます。

それでは，長いことお待せ致しました。もう，あなたは第100図のプログラムを理解することができますよ。まず，**第104図**を御覧ください。これが，その**フローチャート**です。図のように，**三つの部分**で構成されていることがわかります。以下，このフローチャートにしたがって見ていくことにします。

(Ｉ) **転送バイト数を求める**

① キャリー・フラグを０にする。

```
XOR  A
```

② DEレジスタに転送前プログラムの先頭番地をセットする。

```
LD   DE, 0D000H
```

③ HLレジスタに転送前プログラムの最終番地をセットする。

```
LD   HL, 0D064H
```

④ 2バイト減算により，

(最終番地) − (先頭番地)
‖　　　　　‖
HL　　　　DE

の計算をする。結果は，**HLレジスタ**に求まる。

```
SBC  HL, DE
```

⑤ その結果を＋１すれば，**転送バイト数**が求まる。

```
INC  HL
```

(II) **条件のセット**

① 条件のセットとは，**LDIR命令を使うための**準備である

BC←転送バイト数
HL←転送前の先頭番地
DE←転送後の先頭番地

の三条件を各レジスタにセットすることである。

② まず現在，HLレジスタに入っている**転送バイト数**を，BCレジスタに移す。転送は，**スタック領域**を使って行う。

```
PUSH  HL   ……スタック領域へ
POP   BC   ……取り出す
```

第104図　三部構成

③ HLレジスタに転送前プログラムの先頭番地をセットする。

```
LD  HL, 0D000H
```

④ DEレジスタに転送先の先頭番地をセットする。

```
LD  DE, 0E000H
```

(III) **ブロック転送**

これで全ての準備が終っているので，あとはブロック転送命令を実行するだけである。

```
LDIR
```

自分自身をブロック転送する

以上が，第100図の解析です。

解析を終えて，このプログラムが**汎用性のある**ことがおわかりだと思います。すなわち，

① 第100図のプログラムを**SAVE**しておく。

```
WC100, C116↘
```

② もしあるプログラムを他の領域に転送したいことが起きました時には，いま録音しておいたプログラ

ムをロードします。

L↘

③ **Sコマンド**を用い，**第105図**のように

C102H｝転送前プログラム
C103H｝の先頭アドレス
C105H｝転送前プログラム
C106H｝の最終アドレス
C10DH｝転送前プログラム
C10EH｝の先頭アドレス
C110H｝転送後プログラム
C111H｝の先頭アドレス

の**8バイト**を書き換えます。こう書きますと，何かえらく面倒そうに見えますが，実際にやってみると，そう大変なことではありません。

なおこの時，**80系CPUの特徴**を守り，

｛C102H←─下位バイト
｛C103H←─上位バイト

のようにすることに御注意ください。

```
                    ;===================
                    ;  MOVE PROG
                    ;    (81年 10月 31日)
                    ;===================
                    ;
                            ORG   0C100H
                    ;
5C66                MON:    EQU   5C66H        ;GOTO MONITOR
                    ;
C100 AF             MAIN:   XOR   A            ;CY=0
C101 1100D0                 LD    DE,0D000H    ;MINI START
C104 2164D0                 LD    HL,0D064H    ;MINI END
C107 ED52                   SBC   HL,DE
C109 23                     INC   HL           ;HL=BYTES
C10A E5                     PUSH  HL
C10B C1                     POP   BC           ;BC=BYTES
C10C 2100D0                 LD    HL,0D000H    ;FROM
C10F 1100E0                 LD    DE,0E000H    ;TO
C112 EDB0                   LDIR               ;MOVE
C114 C3665C                 JP    MON

                            END
```

第105図　8バイトの書き換え

④ ブロック転送プログラムを走らせます。

GC100↘

```
*SC102
C102 00-    D0-C1 21-   64-16 D0-C1 ED-
C108 52-    23-   E5-   C1-   21-   00-   D0-C1 11-
C110 00-○   E0-D0 ED-○
*
```

第106図　Sコマンドを使って

ところで，この第105図（第100図）のプログラム自身リロケータブルですから，メモリ上の空きエリアでしたら，どこにでも移すことができます。しかも，

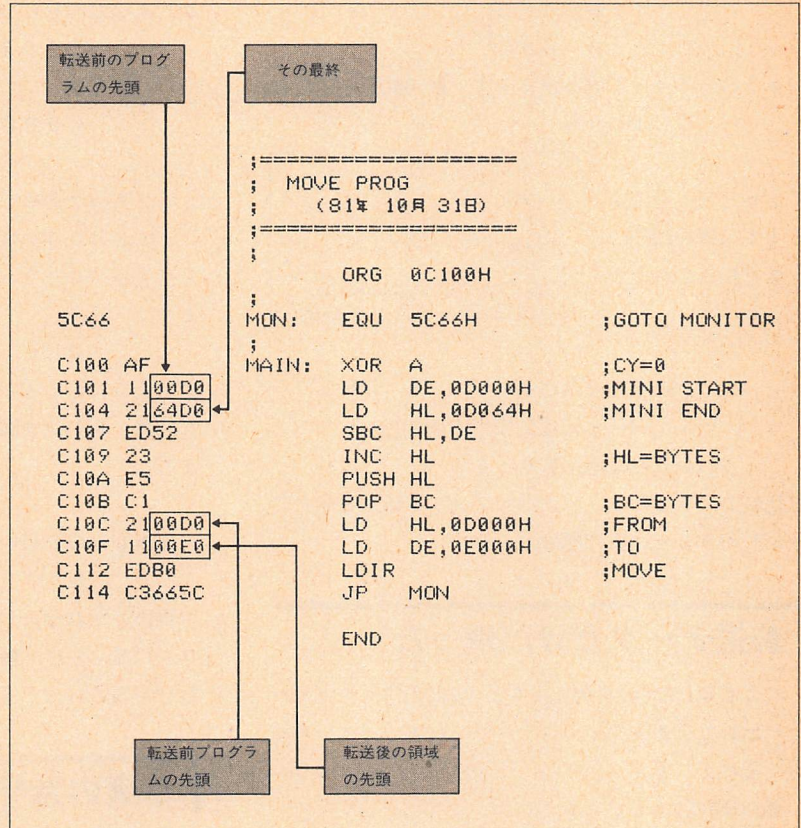
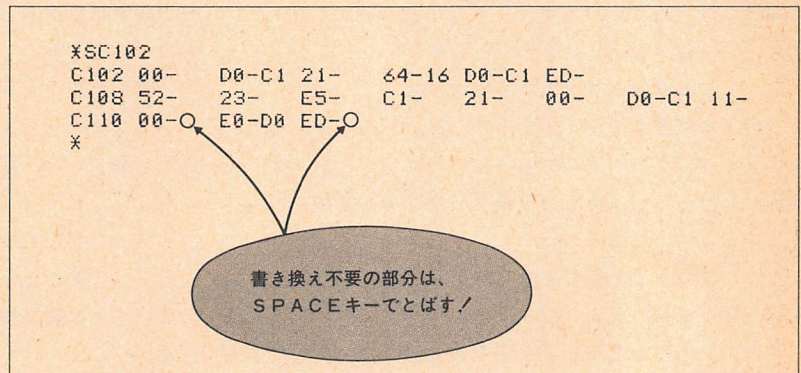
自分自身で自分自身をブロック転送可能

です。だからこそ，非常に便利なプログラムなのです。

それでは実際に，自分自身をブロック転送してみましょうか？　たとえば，今

C100H～C116H

に入っている第105図の"ブロック転送プログラム"をD000Hに移してみることにします。この場合は，

転送前の先頭番地＝C100H
転送前の最終番地＝C116H
転送後の先頭番地＝D000H

ですから，Sコマンドを用いて，

C102H＝00H］
C103H＝C1H］
C105H＝16H］
C106H＝C1H］
C10DH＝00H］
C10EH＝C1H］

C110H＝00H
C111H＝D0H

のように8バイトを書き換えます（**第106図**）。あとは，

GC100↘

でプログラムを走らせます（**第107図**）。うまく転送されたかどうか，

```
*GC100
*■
```

第107図　ブロック転送ON

DC100，C116↘
↘　（←これは見やすくするため）
DD000，D016↘

で比較してみましょう。うまくブロック転送されたことがわかります（**第108図**）。

転送データだけが残った

ここで第102図の説明に戻ります。え？何をやっていたか忘れてしまった？　スイマセンねエ。いろいろ横道にそれまして，

第102図では，もともと

D000H～D064H

の領域で開発された〝ミニ・レジスタ表示プログラム〞を，〝ブロック転送プログラム〞を用いて

E000H～E064H

に**そのまま転送**してみたところでした。ところが，〝ミニ・レジスタ表示プログラム〞は，内部にデータ（文字列）を持っているので，**リロケータブルではない**と考えられます。にもかかわらず，それを

ブロック転送しても意味がない！

という疑問を持ったわけです。

そこで次のような実験をしてみることにします。なお，あなたのPC-8001には第102図のように

D000H～D064H
ミニ・レジスタ表示プログラム
E000H～E064H

それを単純にブロック転送したものとして実験を進めていきます。

まず，D000H～D064Hにある〝ミニ・レジスタ表示プログラム〞を，**Sコマンド**を使って消去します。一応，

DD000，D064↘

で消去されたことを確認します（**第109図**）。これで**〝ミニ・レジスタ表示プログラム〞は，使えなくなりました。**

今あなたのPC-8001には，

E000H～E064H

の得体の知れないデータ（←ブロック転送されたデータ。以後，仮に〝**転送データ**〞と呼ぶことにします）だけが残されています。

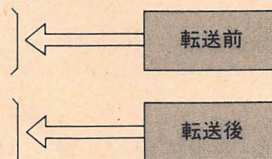
準備の意味は

次に我々が実験することは，この〝転送データ〞を**一つのプログラム**と見なし（しかも，〝ミニ・レジスタ表示プログラム〞であるとみなし），何が起こるか実際に動かしてみようということです。

〝ミニ・レジスタ表示プログラム〞を使うには，**準備**が必要でした。覚えていますか？

F1E3H＝C3H
F1E4H＝00H
F1E5H＝D0H

の**3バイトを書き換えてから使用する**のでしたね？OK？

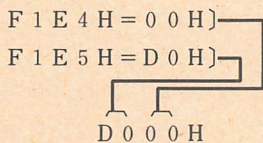
```
*DC100,C116
C100 AF 11 00 C1 21 16 C1 ED 52 23 E5 C1 21 00 C1 11
C110 00 D0 ED B0 C3 66 5C
*
*DD000,D016
D000 AF 11 00 C1 21 16 C1 ED 52 23 E5 C1 21 00 C1 11
D010 00 D0 ED B0 C3 66 5C
*
```

第108図　転送の確認

```
*SD000
D000 FD-00 E5-00 DD-00 E5-00 E5-00 D5-00 C5-00 F5-00
D008 CD-00 CA-00 5F-00 06-00 25-00 21-00 00-00 00-00
D010 39-00 2B-00 2B-00 F9-00 E1-00 11-00 35-00 00-00
D018 19-00 7E-00 CD-00 57-00 02-00 23-00 10-00 F9-00
D020 CD-00 CA-00 5F-00 06-00 06-00 E1-00 CD-00 C0-00
D028 5E-00 CD-00 D4-00 5F-00 10-00 F7-00 E1-00 2B-00
D030 CD-00 C0-00 5E-00 CD-00 D4-00 5F-00 21-00 00-00
D038 00-00 39-00 CD-00 C0-00 5E-00 C3-00 66-00 5C-00
D040 41-00 46-00 20-00 20-00 20-00 42-00 43-00 20-00
D048 20-00 20-00 44-00 45-00 20-00 20-00 20-00 48-00
D050 4C-00 20-00 20-00 20-00 49-00 58-00 20-00 20-00
D058 20-00 49-00 59-00 20-00 20-00 20-00 50-00 43-00
D060 20-00 26-00 20-00 53-00 50-00 99-
*
*DD000,D064
D000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
D060 00 00 00 00 00
*
```

第109図　〝ミニ・レジスタ表示プログラム〞を消去する

ところでＦ１Ｅ４Ｈの**００Ｈ**とＦ１Ｅ５Ｈの**Ｄ０Ｈ**の二つのデータですが，

Ｆ１Ｅ４Ｈ＝００Ｈ〕

Ｆ１Ｅ５Ｈ＝Ｄ０Ｈ〕

Ｄ０００Ｈ

のように並べ換えると，

Ｄ０００Ｈ

という**２バイトのデータ**が得られます。この値，どこかで見たことがありますよね？　そうです。

〝ミニ・レジスタ表示プログラム〞の

スタート・アドレス

が，Ｄ０００Ｈでした。さらに

Ｆ１Ｅ３Ｈ＝Ｃ３Ｈ

のＣ３Ｈとは何でしょう？　付録の〝機械語←→ニーモニック対応表〞を見ますと，

Ｃ３Ｈ＝JP　ｎｎ

となっています。すると，

Ｃ３　００　Ｄ０

の３バイトは，

JP　０Ｄ０００Ｈ

と逆アセンブルされます。これは，

〝ミニ・レジスタ表示プログラム〞の

先頭アドレスにジャンプせよ！

という意味です。すなわち，〝ミニ・レジスタ表示プログラム〞使用前の準備とは，

Ｆ１Ｅ３Ｈ～Ｆ１Ｅ５Ｈ

の３バイトを，〝ミニ・レジスタ表示プログラム〞にジャンプするように書き換えることだったのです。

華麗なる誤解

そこで，

Ｅ０００Ｈ～Ｅ０６４Ｈ

にある〝転送データ〞を走らせる準備をします。前は，

JP　０Ｄ０００Ｈ

になるようにＦ１Ｅ３Ｈ～Ｆ１Ｅ５Ｈを書き換えました。したがって今度は，

JP　０Ｅ０００Ｈ

のように書き換えるのが妥当であると思われます。すると，**Ｓコマンド**を使って

Ｆ１Ｅ３Ｈ＝Ｃ３Ｈ

Ｆ１Ｅ４Ｈ＝００Ｈ

Ｆ１Ｅ５Ｈ＝Ｅ０Ｈ

のように書き換えます(**第110図**)。

```
*SF1E3
F1E3 C9-C3 77-00 00-E0 C9-
*
```

第110図　３バイトの書き換え

これで〝転送データ〞を走らせる準備はできました。さっそく走らせてみましょう。しかし，そのためには**何か別のプログラム**が必要です。なぜなら，そもそも〝ミニ・レジスタ表示プログラム〞は，**自身を直接走らせる**のではなく，何かプログラムを開発中，そのプログラムの途中でレジスタの値を見たい時，

STOP させる位置にＦＦＨを

書き込んで使う

ものでした。

そこでいま，

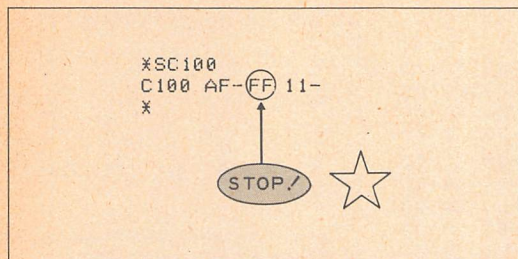
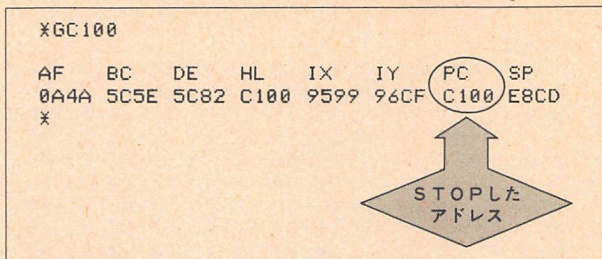
Ｃ１００Ｈからプログラムを開発中

であると仮定し，いきなりＣ１００ＨでプログラムをSTOP させることにします。それには，**Ｓコマンド**を使って

第111図　C100HでSTOPさせる

```
¥GC100

AF   BC   DE   HL   IX   IY   PC   SP
0A4A 5C5E 5C82 C100 9599 96CF C100 E8CD
¥
```

STOPしたアドレス

第112図　転送データが走った

C100H＝FFH

と書き込んでやれば良いのです(**第111図**)。

さあ，すべてはOKです。いよいよプログラムを走らせますよ。

GC100↘

結果は，――。何とレジスタの値が表示されたではありませんか(**第112図**)。**プログラム・カウンタ**（PC）の値に御注目ください。ちゃんと，

PC＝C100H

になっています。これは，プログラムがきちんと

C100H
↑
FFHを書き込んだ番地

で STOP したことを示しています。

さて，大変なことになりました(マア，それ程たいしたことではないのですが)。本来

リロケータブルではない！

はずの〝ミニ・レジスタ表示プログラム〟が，何と

リロケータブルであった！

のです。

まとめ

ところで，そもそも〝ミニ・レジスタ表示プログラム〟は，

リロケータブルであるべき

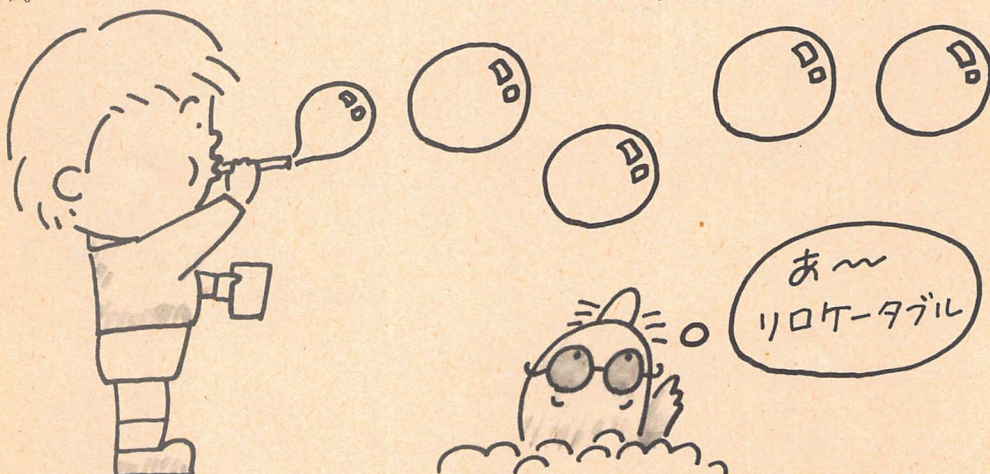
性質のものです。なぜならそれは，

プログラム開発支援用

のプログラムで，プログラム開発の際に利用する**道具**です。

もし〝ミニ・レジスタ表示プログラム〟が，リロケータブルでなかったら，どうでしょうか？　普段は，特に支障はないかもしれません。しかし，もし

開発中のプログラムの占める領域
〝ミニ・レジスタ表示〟の占める領域

が重なったらどうしますか？　もはやその時は，〝ミニ・レジスタ表示プログラム〟を使用することはできません。しかし，その〝ミニ・レジスタ…〟がリロケータブルであれば，すかさず〝ブロック転送プログラム〟を用いて**他の領域に転送**した上で使用することができます。

```
                 ;==================================
                 ;  MINI REGISTER DISPLAY
                 ;     (81年 3月 18日):BY K.TSUKAGOSHI
                 ;==================================
                 ;
                        ORG   0FF40H
                 ;
0257             OCRT:  EQU   257H           ;OUT CRT
5C66             GMON:  EQU   5C66H          ;GOTO MONITOR
5EC0             PRHL:  EQU   5EC0H          ;PRINT HL
5FCA             CRLF:  EQU   5FCAH
5FD4             PSPC:  EQU   5FD4H          ;PRINT SPACE
                 ;
FF40 FDE5        MAIN:  PUSH IY              ;REGISTER STORE FOR DISPLAY
FF42 DDE5               PUSH IX
FF44 E5                 PUSH HL
FF45 D5                 PUSH DE
FF46 C5                 PUSH BC
FF47 F5                 PUSH AF
                 ;
FF48 CDCA5F             CALL CRLF            ;CARRIAGE LINE FEED,PREPRA FOR DISPLAY
                 ;
FF4B 0625               LD   B,37            ;'AF-SP'=37 CHARACTER
FF4D 210000             LD   HL,0            ;LET HL=SP
FF50 39                 ADD  HL,SP
FF51 2B                 DEC  HL
FF52 2B                 DEC  HL
FF53 F9                 LD   SP,HL
FF54 E1                 POP  HL
FF55 113500             LD   DE,53           ;HL=ADR(DATA)
FF58 19                 ADD  HL,DE
                 ;
FF59 7E          MA1:   LD   A,(HL)          ;PRINT 'AF-SP'
FF5A CD5702             CALL OCRT
FF5D 23                 INC  HL
FF5E 10F9               DJNZ MA1
                 ;
FF60 CDCA5F             CALL CRLF
                 ;
FF63 0606               LD   B,6             ;6 REGISTERS
FF65 E1          MA2:   POP  HL
FF66 CDC05E             CALL PRHL
FF69 CDD45F             CALL PSPC
FF6C 10F7               DJNZ MA2
                 ;
FF6E E1                 POP  HL              ;PRINT PC
FF6F 2B                 DEC  HL
FF70 CDC05E             CALL PRHL
FF73 CDD45F             CALL PSPC
                 ;
FF76 210000             LD   HL,0            ;PRINT SP
FF79 39                 ADD  HL,SP
FF7A CDC05E             CALL PRHL
                 ;
FF7D C3665C             JP   GMON            ;GOTO MONITOR
                 ;
FF80 41462020    DATA:  DB   'AF   BC   DE   HL   '
FF84 20424320
FF88 20204445
FF8C 20202048
FF90 4C202020

FF94 49582020           DB   'IX   IY   PC   SP'
FF98 20495920
FF9C 20205043
FFA0 20202053
FFA4 50
                 ;
                        END
```

第113図　理想的な位置に

以上の理由により，私は〝ミニ・レジスタ表示プログラム〟を開発するにあたり

　　　リロケータブル

になるように設計したのです。その際，確かに

　　内部にデータ領域があり，そこをレジスタ・

　　ペアで示さなければならない

という問題はありました。しかし，マアそこは適当に解決していますので御安心ください。第99図でいえば

　　　D00BH～D018H

の部分です。ここは，リロケータブルを保ちつつ，もう少し**短くする**こともできます。暇があったら挑戦してみてください。

以上，長らく〝ミニ・レジスタ表示プログラム〟にまつわるリロケータブルの話題を御紹介してきました。ここで，その使い方を右にまとめておきます。

以上が，〝ミニ・レジスタ表示プログラム〟についてのまとめです。ところで，この〝ミニ・――〟，どの領域にでも置くことが可能ですが，それなら一体

　　　どこに置くのが理想的？

でしょうか？

それは，できるだけ他のプログラムと重ならない領域が良いですね。たとえば，**第113図**にあるように

　　　FF40H～FFA4H

あたりに置くのが良いでしょう。この番地は，**普通は使用されていません**。もしそれでも重なるようでしたら，その時初めてリロケートすれば良いのです。

<ミニ・レジスタ表示プログラム>

機　能：ユーザー開発中のアプリケーション・プログラムにおいて，任意のアドレスでプログラムの実行を中断し，その時のレジスタの内容を表示する。

使用前の準備：

使用前に本プログラムをメモリにロードしたら，F1E3H～F1E5Hの3バイトを

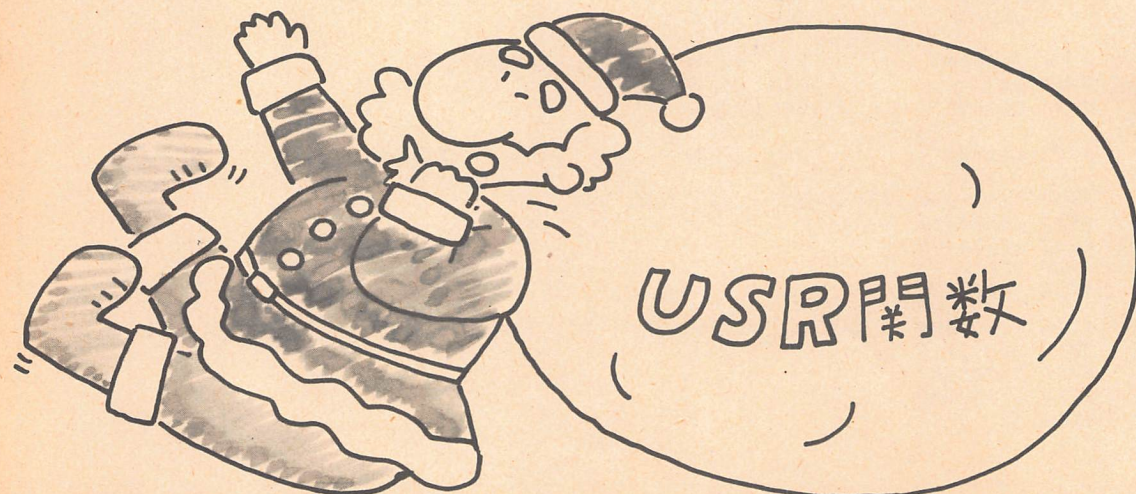
　　JP　nn
　　　　↑
　　　　本プログラムの先頭番地

になるように書き換える。具体的には，

　　F1E3H＝C3H
　　F1E4H＝番地の下位バイト
　　F1E5H＝番地の上位バイト

のようにする。

使用法：アプリケーション・プログラム（開発中の応用プログラム）のSTOP させたい番地に，FFHを書き込む。その後，アプリケーション・プログラムを実行すると所期の目的が果せられる。

備　考：本プログラムはリロケータブルであるので，任意の領域で走らせることが可能である。

第9章

USR関数

```
DC00: CD 95 D9 E6 07 28 17 FE 03 38 0E FE 05 38 05 21 ; 060F
DC10: 32 00 18 0D 21 64 00 18 08 21 96 00 18 03 21 2C ; 021B
DC20: 01 E5 ED 5B CD E3 1C CD B6 D9 E1 ED 4B C6 E3 09 ; 0A21
DC30: 22 C6 E3 C9 CD 3F DC 21 00 00 22 AD E3 E1 C9 FE ; 08F7
DC40: 8C 28 26 FE C8 28 22 FE 42 28 1E FE 77 28 1D FE ; 0728
DC50: 7F 28 19 FE 96 28 15 FE F7 28 11 FE EF 28 11 FE ; 07E3
DC60: FE 28 0D FE FF 28 09 E1 C9 AF 18 06 3E 42 18 02 ; 0672
DC70: 3E 96 77 47 C9 3E 78 2E 12 06 05 F5 C5 E5 CD 60 ; 0728
DC80: D9 E1 C1 F1 2C 10 F4 C9 FE 5A C0 21 00 00 22 AD ; 086D
DC90: E3 E1 C9 3E FF 32 B3 E3 21 00 00 22 C8 E3 C9 CD ; 0916
DCA0: 32 D0 CD 64 DD CD E2 D8 3E 0A 32 B2 E3 21 00 00 ; 07C7
DCB0: 22 C6 E3 22 CA E3 CD 97 D8 3E 03 32 D0 E3 AF 32 ; 08DD
DCC0: E6 E3 C9 CD E2 D8 CD 97 D8 21 B2 E3 7E FE 0F 38 ; 0ACE
DCD0: 02 36 0A CD 70 D7 CD 11 DD 3E FF 32 B9 E3 21 CA ; 0807
DCE0: E3 34 CD 8E D8 CD 75 DC CD B2 D8 CD BD DA AF 32 ; 0B04
DCF0: D1 E3 21 00 00 22 CD E3 22 AD E3 22 D2 E3 C9 3E ; 0837
DD00: 28 32 AC E3 AF 32 AA E3 32 AF E3 CD 4A DD C3 B3 ; 0985
DD10: D4 3A B2 E3 D6 03 6F 11 A8 DF C3 CC D7 3A E6 E3 ; 09EC
DD20: A7 C0 2A C6 E3 11 DC 05 CD D3 5E D8 21 D0 E3 34 ; 090A
DD30: 21 E6 E3 7E 2F 77 06 32 C5 01 20 15 CD FF D9 06 ; 06EC
DD40: 08 CD 73 D9 C1 10 F1 C3 4A DD 21 00 41 CD 0C D9 ; 07E1
DD50: E5 06 03 36 00 23 10 FB E1 06 EA 3A D0 E3 3D C8 ; 0715
DD60: 70 23 18 FA CD E2 D8 AF D3 51 11 FF 22 21 89 E1 ; 08BC
DD70: CD 37 DA 06 28 C5 01 1A 2D CD FF D9 CD 85 D9 C1 ; 08AA
```

べーしっく　と　ましんご　の　りんく……

道具を使う

第９章の初めに，**最後の懸案事項**をかたづけておきます。

第114図を御覧ください。これは第69図と同じアセンブル・リストです。我々はまだこのプログラムを走らせていません。リストのＣ１１２Ｈを御覧ください。

理論上は(誤りがなければ)，この時点で

　　ＨＬ＝０Ａ７２Ｈ

となっているはずです。そして

　　ＪＰ　(ＨＬ)

でＡ７２番地にジャンプさせるはずでした。しかし，

　　ＨＬ＝０Ａ７２Ｈ

である自信がなかったので，**プログラムを走らせるのを躊躇していたのでした。**

```
                    ;===================
                    ;  KEY LIST
                    ;   (81年 10月 23日)
                    ;===================
                    ;
                            ORG   0C100H
                    ;
0081                NBAS:   EQU   81H
                    ;
C100 218100         MAIN:   LD    HL,NBAS
C103 E5                     PUSH  HL
C104 3A13C1                 LD    A,(DATA)
C107 47                     LD    B,A
C108 0E0A                   LD    C,10
C10A C5                     PUSH  BC
C10B D1                     POP   DE
C10C 7A                     LD    A,D
C10D 53                     LD    D,E
C10E 5F                     LD    E,A
C10F EB                     EX    DE,HL
C110 23                     INC   HL
C111 23                     INC   HL
C112 E9                     JP    (HL)      ← HL=?
                    ;
C113 70             DATA:   DB    70H
                    ;
                            END
```

第114図　最後の懸案リスト

いまや我々は，プログラムの実行を途中でSTOPし，

レジスタの値を見る

ための道具を知っています。そうです。〝ミニ・レジスタ表示プログラム〟の使用法を長らく見てきました。さっそくこれを利用し，**懸案事項をかたづける**ことにします。

```
¥SF1E3
F1E3 C9-C3 77-40 00-FF C9-
¥
```

第115図　使用の準備

```
¥DC100,C113
C100 21 81 00 E5 3A 13 C1 47 0E 0A C5 D1 7A 53 5F EB   } 実験用プログラム
C110 23 23 E9 70
¥
¥DFF40,FFA4
FF40 FD E5 DD E5 E5 D5 C5 F5 CD CA 5F 06 25 21 00 00   }
FF50 39 2B 2B F9 E1 11 35 00 19 7E CD 57 02 23 10 F9   }
FF60 CD CA 5F 06 06 E1 CD C0 5E CD D4 5F 10 F7 E1 2B   }
FF70 CD C0 5E CD D4 5F 21 00 00 39 CD C0 5E C3 66 5C   } ミニ・レジスタ表示
FF80 41 46 20 20 20 42 43 20 20 20 44 45 20 20 20 48   } プログラム
FF90 4C 20 20 20 49 58 20 20 20 49 59 20 20 20 50 43   }
FFA0 20 20 20 53 50                                    }
¥
```

第116図　２つのプログラムをロードして

まず〝ミニ・レジスタ表示プログラム〟をLOADします（ここでは，FF40H～FFA4Hにリロケートしてあります）。使用前の準備

F1E3H～F1E5H

の書き換えもしておきましょう（**第115図**）。第114図のプログラムも入っていますね？　**第116図は，**

C100H～C113H

実験用プログラム（第114図）

FF40H～FFA4H

ミニ・レジスタ表示プログラム

が入力されていることを**Dコマンド**で確認したところです。

――プログラムを**STOP**させたい番地は？

――C112H（第114図参照）

OK！　そこに**Sコマンド**で

FFH

を書き込みましょう（**第117図**）。準備完了。**プログラム発射！**

GC100↘

第118図のようにレジスタの値が表示されました。**PC（プログラム・カウンタ）**の値を見ると，

PC＝C112H

となっています。ちゃんと予定の位置に止まっています。そして，**HLレジスタの値**に注目しましょう。

HL＝0A72H

になっています。どうやら理論どおりに動いたようです。

第117図　FFをセットする

第118図　プログラムGO！

プログラムの正体

プログラムはうまく動いたようです。これで安心して本番に移れます。書き換えておいたC112Hを

C112H＝E9H

に戻します（**第119図**）。そして改めて

GC100↘
でプログラム・スタートです。結果は，**写真4**のとおりです。これ何だか知っていますよね？ 同じものは，**BASIC のコマンド・レベル**で

KEYLIST↘

とキー・インしても得られます（**写真5**）。これは現在の

ファンクション・キーの内容

を表示したものです。実は，PC-8001のROMの中には，A72番地から**KEYLISTの処理ルーチン**が書かれています。第114図のプログラムは，このルーチンをCALLするものだったのです。

```
*SC112
C112 FF-E9 70-
*
```

第119図　プログラムを元に戻して

ここで第114図のプログラムについて，二点程**補足**をしておきます。

その一つ目。第114図のリストにおいて最初の

```
LD      HL, NBAS
PUSH    HL
```

の意味が良く理解できなかったかもしれません。特にこのプログラムにおいて，この4バイトが不要に見えたかもしれません。しかし，この4バイトは必要なのです。ちなみにこの4バイトを

00H

に書き換えてみましょう（**第120図**）。00Hは，“Z―80”の命令表を見ればおわかりのように，**何の機能も持ちません**。したがってプログラムの実行中，

00Hの部分は無視

されます。この上で

GC100↘

とやってみましょう。今後は，**ベルが鳴り**，↘

第120図　プログラムを消去すると

```
Ok
mon
*SC100
C100 00-21 00-81 00-00 00-E5
C104 3A-3A 13-13 C1-C1 47-47
C108 0E-0E 0A-0A C5-C5 D1-D1
C10C 7A-7A 53-53 5F-5F EB-EB
C110 23-2C 23-2C E9-E9 70-70
C114 CD-
*GC100
```

《写真4》GC100でプログラムスタート

《写真5》KEYLISTでも可能

《写真6》SYNTAX ERRORが出た

syntax error

が表示されてしまいました（**写真6**）。これは，SP（スタック・ポインタ）の値がうまく合わなかったためです。詳しくは説明しませんが，

C100H～C103H

の4バイトは，そのSPの値を調整し，うまく**BASIC**

に戻す働きをさせています。

補足のその2。以前，第68図のプログラムを作るにあたり，わざとミスると申し上げておきました。どこが誤りかお気付きになりましたか？

答は，C110HとC011Hの

INC　HL
INC　HL

です。ここは，〈チャレンジ〉の指示が，

「Lレジスタの値を＋2する」

ということでした。

Lレジスタの値を増加させる命令
⇨　INC　L

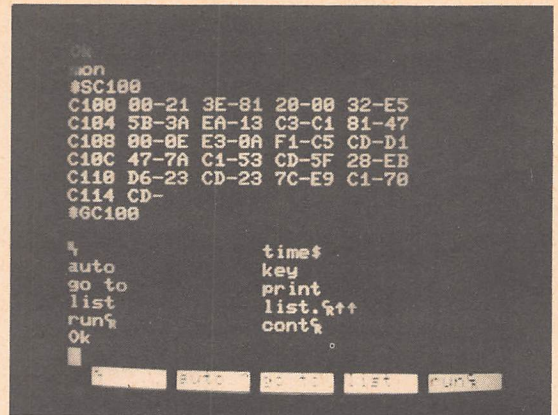
です。

INC　HL

ではありません。したがってこの2バイトは，

INC　L
INC　L

とすべきでした。第114図のプログラムが正しく動いたのは，たまたま**ケガの功名**だったのです。いつもいつもミスったままプログラムが正常に動くとは限りません。しかし，そんな時でも〝ミニ・レジスタ表示プログラム〟を使えば，プログラムが正しく動いているかをチェックすることができます。どうか〝**ミニ・――〟を有効に御活用ください**。

念のために第114図のプログラムを正しくなおしたものを，**第121図**に掲げておきます。**第122図**が，そのダンプ・リストです。もちろんプログラムは，正常に動きます（**写真7**）。

《写真7》ダンプ・リスト入力・実行

BASICとマシン語

ここから話しは，一気に**飛躍**していきます。かつ，話しは**具体化**して行きます。そこには，あなたの知りたかった概念が次から次へと登場してくることでしょう。期待しつつ，読み進めていってください。

```
                  ;====================
                  ;  KEY LIST
                  ;    (81年 10月 23日)
                  ;====================
                  ;
                           ORG   0C100H
                  ;
0081              NBAS:    EQU   81H          ;N-BASIC HOT START
                  ;
C100 218100       MAIN:    LD    HL,NBAS
C103 E5                    PUSH  HL           ;RET=N-BASIC
C104 3A13C1                LD    A,(DATA)
C107 47                    LD    B,A
C108 0E0A                  LD    C,10         ;BC=700AH
C10A C5                    PUSH  BC
C10B D1                    POP   DE           ;DE=700AH
C10C 7A                    LD    A,D
C10D 53                    LD    D,E
C10E 5F                    LD    E,A          ;DE=0A70H
C10F EB                    EX    DE,HL        ;HL=0A70H
C110 2C                    INC   L
C111 2C                    INC   L            ;HL=0A72H
C112 E9                    JP    (HL)         ;GOTO 0A72H
                  ;
C113 70           DATA:    DB    70H
                  ;
                           END
```

第121図　正しいプログラム

```
¥DC100,C113
C100 21 81 00 E5 3A 13 C1 47 0E 0A C5 D1 7A 53 5F EB
C110 2C 2C E9 70
¥
```

第122図　ダンプ・リスト

あなたは，**BASIC** と**マシン語**を比べてどう思いますか？

「マシン語は，速い」

「BASICは，遅い」

「マシン語は，開発するのに手間がかかる」

「BASICは，手軽に開発できる」

いろいろな意見が出ますね。要約すると，大体次のようなことになるでしょう。

BASIC

(長所)：生産性が高い。

(短所)：実行速度が遅い。

マシン語

(長所)：実行速度が速い。

(短所)：生産性が低い。

以上のように，BASIC，マシン語それぞれが良い点，悪い点を持っています。ところで，一般にプログラムの中身を分解すると，

{ かなり**スピードを要求**される部分

{ それ程**スピードには関係のない**部分

の二つに分かれます。それなら

スピードを要求される部分

だけをマシン語で組み，その他はBASICで組んだらどうでしょうか？　かなり合理的だと思いませんか？すなわちBASIC，マシン語のそれぞれ良い所を取り入れ，できるだけ手抜きをし，

> 可能な限りBASICを使って組む。しかし，やむを得ずスピードを要求されるところだけは，マシン語を使って組む。

方針で開発に当たると，**効率良く**プログラムの開発を行うことができます。

USR関数の登場

普通，このような設計方針を立てた時，

メイン・ルーチン

サブルーチンの大部分

は**BASIC**を使って記述します。そして最少限必要な部分だけを

マシン語のサブルーチン

で作り，その部分を**BASIC側からCALL**して使います。そんな時に用いるBASICの関数が，かの有名な（たぶんあなたの苦手な）

ＵＳＲ関数

です。

これからしばらくは，この

ＵＳＲ関数のマスター

が目標です。

ＵＳＲ関数の最も簡単な使い方から御紹介致しましょう。ここでもＰＣ-8001のＲＯＭ内サブルーチンに登場してもらいます。マア，何でも良いのですが，

> **ＢＥＥＰ**
> **〈アドレス〉**：Ｄ４３Ｈ
> **〈機　　能〉**：一定時間ブザーを鳴らす

を使うことにします。この

マシン語のサブルーチン

を**BASICから呼び出してみよう**というわけです。

方法は，まず**マシン語サブルーチンの先頭アドレス**Ｄ４３Ｈを**ユーザー関数にセット**することから始めます。それには，

ＤＥＦ　ＵＳＲ

という命令を用い，

ＤＥＦ　ＵＳＲ＝&ＨＤ４３
（↑先頭番地）

（&Ｈは16進数を表わす）

これでＵＳＲ関数のアドレスが定義されましたので，あとはこのサブルーチンを呼び出すだけです。それには，いろいろな方法がありますが，たとえば

Ｘ＝ＵＳＲ（１）

のように代入文を実行すれば良いのです。ここで

左辺の変数———ここではＸ

（　）の中の数——ここでは１

は何であってもかまいません。

でき上がったプログラムが，**第123図**です。これは，

ＢＡＳＩＣ

のプログラムですから，単に

ＲＵＮ↘

とすれば実行できます。ピーという**ビー**

```
10 DEF USR=&HD43
20 X=USR(1)
```

第123図　ＵＳＲ関数を使う

プ音が聞こえたでしょう？

複数のUSR関数

以上が，もっとも基本的なＵＳＲ関数の使い方です。まとめておきますと，

① ＤＥＦ　ＵＳＲでマシン語サブルーチンの先頭アドレスを定義しておく。
② **代入文**を実行する。これでマシン語のサブルーチンがCALLされる。

さて，たぶん気が付いたことと思いますが，このやり方では，

一つのマシン語サブルーチン

しか使えません。実際は，**複数のマシン語サブルーチン**をCALLするのが普通です。そんな時のためにN－BASICでは，

10個のＵＳＲ関数

が用意されています。それらは，ＵＳＲのあとに0～9までの数をつけて区別しています。すなわち

ＵＳＲ０
ＵＳＲ１
ＵＳＲ２
　⁝
ＵＳＲ９

の10個です。そして，それぞれのＵＳＲ関数に**別のサブルーチンのアドレス**を定義して用いることができます。

それでは，具体的に複数のサブルーチンを使ってみましょう。ここで用いるマシン語サブルーチンは，次の三つです。

CLEAR
　〈アドレス〉：４５ＡＨ
　〈機　　能〉：画面をクリアする。
KEY LIST
　〈アドレス〉：Ａ７２Ｈ
　〈機　　能〉：キーリストを表示する。
BEEP
　〈アドレス〉：Ｄ４３Ｈ
　〈機　　能〉：一定時間ベルを鳴らす。

もう，おなじみのサブルーチンですね？

複数マシン語サブルーチンをCALLする

そこで，

〈チャレンジ〉
　画面をクリアしたあと，キー・リストを表示し，ベルを鳴らすプログラムを作りなさい。

に挑戦してみましょう。

まず，**アドレスの定義**です。

ＤＥＦ　ＵＳＲ０＝＆Ｈ４５Ａ
ＤＥＦ　ＵＳＲ１＝＆ＨＡ７２
ＤＥＦ　ＵＳＲ２＝＆ＨＤ４３

で三つのマシン語サブルーチンの先頭アドレスをセットします。これで三つともＵＳＲ関数を使用できるようになりましたから，**代入文**を用いてＵＳＲ関数を実行します。

Ｘ＝ＵＳＲ０（1）　←─画面のクリア
Ｘ＝ＵＳＲ１（1）　←─キー・リスト
Ｘ＝ＵＳＲ２（1）　←─ベル

第124図ができあがったプログラムです。

ＲＵＲ↘

で実行されます。サッと画面がクリアされ，キーリストが表示され，ベルが鳴ってプログラムの実行が停止されます(**写真８**)。

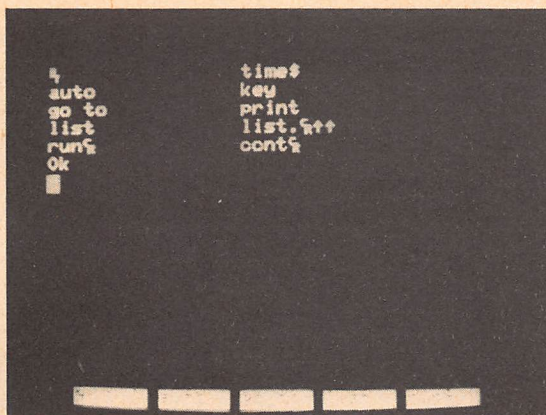
```
10 DEF USR0=&H45A'                  ;clear
20 DEF USR1=&HA72'                  ;key list
30 DEF USR2=&HD43'                  ;beep
40 X=USR0(1):X=USR1(1):X=USR2(1)
```

第124図　複数のＵＳＲ関数を使う

（注１）
　数字のつかないＵＳＲは，ＵＳＲ０と同じです。すなわち，N－BASICでは

USR＝USR0

として扱われます。

（注２）
　ユーザー関数の定義は，そのユーザー関数を使う前でしたら，どこに置いてもかまいません。

《写真８》プログラム停止

（注３）

一度定義さえしておけば，同じＵＳＲ関数を何度でも使うことができます。たとえば，

```
DEF　USR=&HD43
X=USR（1）
X=USR（1）
```

でベルが二回鳴ります。

（注４）

ＵＳＲ関数は全部で10種類ありますが，それでも不足する時は，とりあえず不用なＵＳＲ関数を**再定義**し，他のマシン語サブルーチンを呼び出すようにします。たとえば，

```
DEF　USR8=&H45A
X=USR8（1）
DEF　USR8=&HD43
X=USR8（1）
```

とすれば，ＵＳＲ８を同一プログラム内で画面クリアとＢＥＥＰの二種類のマシン語サブルーチンをCALLするのに使うことができます。

ビーム砲のデザイン

以上，ＵＳＲ関数のもっとも単純な使い方を御紹介してきました。しかし，これだけでは

ＵＳＲ関数の使い方

としては，不十分なのです。もう一つ

引数（パラメータ）の授受

という概念を理解する必要があります。

ここから少し，**話しがこみいってきます**ので，ゆっくりと読み進めていってください。

これから，

ビーム砲を左右に動かす

ということを，

BASIC＋マシン語

でプログラミングしてみようと思います。方針としては，まず

オールBASIC

でプログラムを組んで行きます。しかる後に

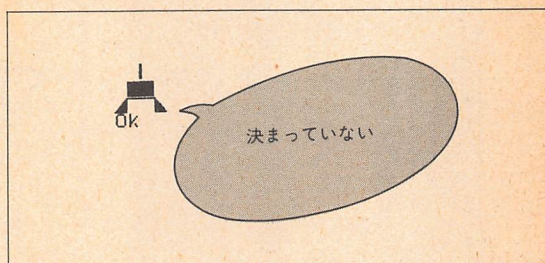
遅い部分をマシン語化する

という方向で作業を進めて行くことにします。

まず，**ビーム砲のデザイン**です。自由にデザインしていただけば結構なのですが，ここでは

ヨコ：４×３：タテ

の大きさで設計してみました。**第125図**のデザインでいかがでしょう？　キャラクタで書けば，

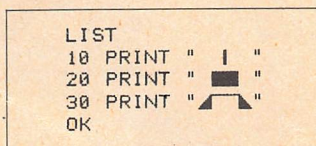
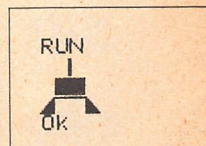
という具合です。

第125図　ビーム砲のデザイン

次にこれを画面に出力しましょう。簡単です。

```
PRINT　"　　"
PRINT　"　　"
PRINT　"　　"
```

でできます(**第126図**)。これを

RUN↘

すれば，**第127図**のようにいま作ったビーム砲が出力されます。

第126図　画面に出力するには

第127図　ＲＵＮでスタート

サブルーチン化する

ところで第126図のプログラムでは，ビーム砲が

カーソルの位置に表示される

だけで，あまり汎用性がありません。そこでこの部分を**サブルーチン化**し，メインルーチンで

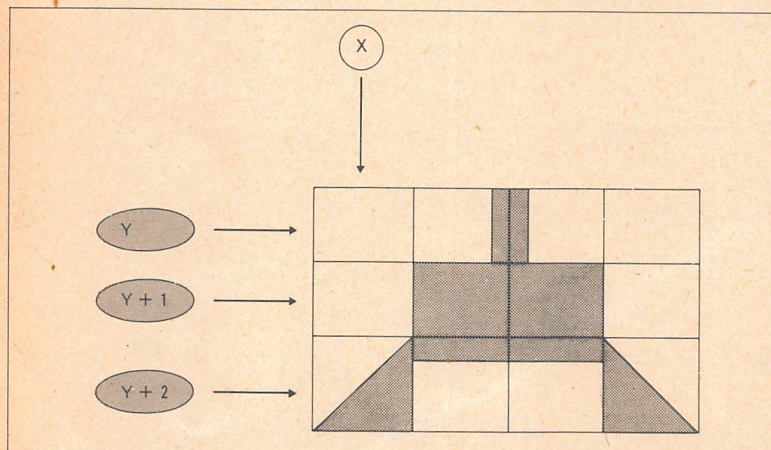
X←―ヨコ座標
Y←―タテ座標

を指定してCALLすると，

LOCATE　X，Y

の位置にビーム砲が表示されるように書き換えてみましょう。ただし，ビーム砲は

4×3

の大きさを持っていますから，

(X，Y)

の位置は，**ビーム砲の左肩**の位置を指すものとします(**第128図**)。

そこで，ビーム砲の各行のタテ座標を変数Yを使って表わすことを考えます。

1行目……Y
2行目……Y＋1
3行目……Y＋2

となりますね(**第129図**)？　したがってこのサブルーチンは，

```
LOCATE　X，Y
  PRINT　"  ▌  "
LOCATE　X，Y＋1
  PRINT　" ■■ "
LOCATE　X，Y＋2
  PRINT　"◢▀▀◣"
RETURN
```

ということになります。ただし，後々のために，ビーム砲の左右に空白(スペース)を設け，

```
PRINT　"△  ▌  △"
PRINT　"△ ■■ △"
PRINT　"△◢▀▀◣△"
```

(△は，スペースを表す)

のようにPRINTすることにします。こうしてできあがったサブルーチンが，**第130図**です。したがって，ビーム砲の**仮の大きさ**は，

ヨコ：6×3：**タテ**

となります。

第128図　左肩の位置を示す

第129図　ビーム砲各行の左端の座標は？

このサブルーチンの使い方を例をあげて示しましょう。たとえば，

LOCATE　10，10

の位置にビーム砲を表示させたいとしたら，メイン・ルーチンで

X＝10：Y＝10

を指定し，このサブルーチンをCALLすれば良いのです(**第131図**)。

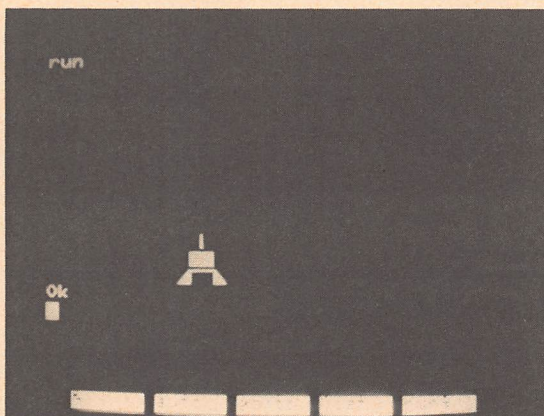
RUN↘

すると，**写真9**のようにビーム砲が表示されます。

```
1120 'PRINT BEAM                        ;IN=X,Y
1130 LOCATE X,Y   :PRINT "  ╻  "
1140 LOCATE X,Y+1:PRINT "  ▄▄  "
1150 LOCATE X,Y+2:PRINT " ◢▀▀◣ "
1160 RETURN
```

第130図　サブルーチンの完成

```
100 X=10:Y=10:GOSUB 1120
110 END
```

第131図　(10, 10)でサブルーチンをコール

《写真９》ビーム砲が表示される

右に動かす

これでビーム砲を表示するサブルーチンができましたので，これを**左右に動かす**ことを考えましょう。まず，右に移動させてみます。

ビーム砲が右（左右）に動いても，

上下の変化はない

＝タテ座標（変数Ｙの値）は一定である

ことに注意してください。そこで先に**Ｙの値**を決めてしまいます。どの行で動かしましょうか？

仮に，

Ｙ＝２０

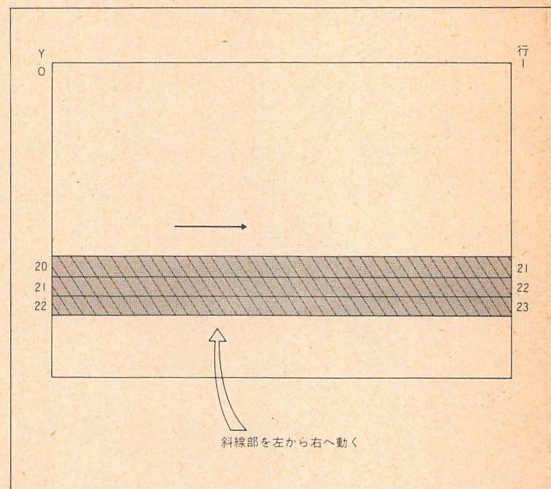
と決めます。こう決めると，ＴＶ画面の一番上が

Ｙ＝０

ですから，ビーム砲は，

20行目……ビーム砲の１行目

21行目……ビーム砲の２行目

22行目……ビーム砲の３行目

第132図　Ｙ座標は

を動くことになります(**第132図**)。

次に**横座標（変数Ｘの値）**を考えます。

ビーム砲を左端から右端まで動かすわけですから，初期値は，

左　端：Ｘ＝０

です。これは，すぐにわかりますね。問題は，右端です。まず**画面のサイズ**により変化します。そこで

８０×２５

のサイズで設計することに致しましょう．すると，

右　端：Ｘ＝７９？

で良いでしょうか？　老婆心ながら

Ｘ＝８０

と考えた人は，問題外ですよ。LOCATE は，**左端を０**として数えますから，80サイズの画面では，右端は，Ｘ＝７９となります。

すると，どうやら

Ｘ＝７９

が正しそうな気がしてきます。しかし，結論を申し上げますと，これは誤りです。

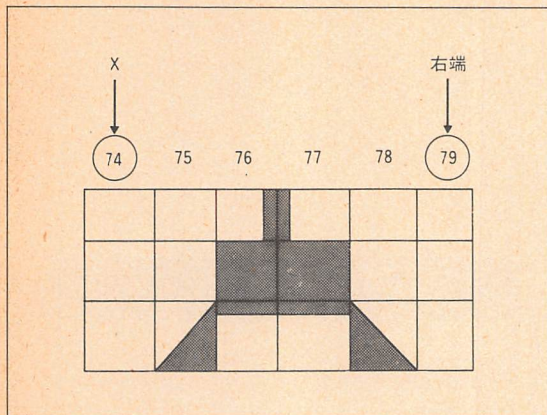
右端の位置は？

なぜなら，ビーム砲には幅があるからで**第133図**のように

右端：Ｘ＝７４

までしか行けません。したがってビーム砲は，

Ｘ＝０　⟶　Ｘ＝７４

の間を動くことになります。

第133図　右端に来た時の状況

次にビーム砲を動かすアルゴリズムを考えます。
これは簡単で，

を繰り返すことで実現できます。**ビーム砲の左右にある空白**で，前の位置のビーム砲が消去されるからです（**第134図**）。

したがってビーム砲を右に動かすアルゴリズムは，

① Yの値を20にする。
② Xの値を０にする。
③ （X，Y）の位置にビーム砲を出力する（先に作ったサブルーチンをCALLすれば良い）。
④ Xの値を一つ大きくする。
⑤ ③に戻る。

となります。そしてこれを，Xの値が

X＝７４

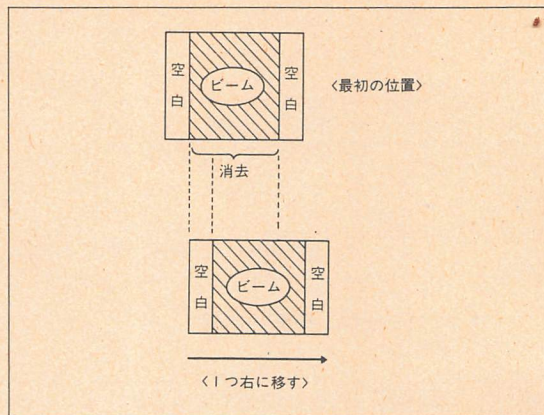
になるまでループさせます。

以上のアルゴリズムは，

FOR～NEXTループ

を用いて簡単に実現することができます。すなわち，

FOR　X＝0　TO　７４
　(X，Y）にビーム砲表示
NEXT

で右への移動が終了します。これを基に，右への移動をプログラミングすると，**第135図**のようになります。

第134図　空白の役割り

```
1060 Y=20'                ;SET LOCATE-Y
1070 FOR X=0 TO 74'       ;MOVE RIGHT
1080   GOSUB 1120
1090 NEXT
```

第135図　右に動かすルーチン

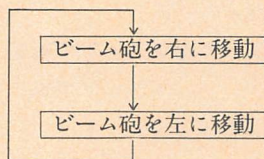
オールBASIC版

右への移動がわかれば，左への移動は簡単です。それを逆にすれば良いからです。

FOR　X＝７４　TO　０　STEP　－１
　(X，Y）にビーム砲表示
NEXT

がそのアルゴリズムです。これをプログラム化し，先程のメイン・ルーチンのあとにつなげてやれば，ビーム砲が，

左端から右端に移動し，また左端へ
戻ってくる

プログラムが出来上がります。さらに

のように**無限ループ**を構成すれば，ビーム砲が

左右に行ったり来たり

するプログラムが作れます。

こうして出来上がったプログラムが，**第136図**です。これを入力して，

RUN↘

させると，かわいらしい(？)ビーム砲が左右に動く様

```
1000 '==================================
1010 '  MOVING BEAM GUN
1020 '       1982.11.7:BY K. TSUKAGOSHI
1030 '==================================
1040 '
1050 WIDTH 80,25:CONSOLE 0,25,0,0'          ;TV MODE SET     ← 画面を80字モードに
1060 Y=20'                                  ;SET LOCATE-Y    ← Yを決める
1070 FOR X=0 TO 74'                         ;MOVE RIGHT      ┐
1080   GOSUB 1120                                            │ 右に動かす
1090 NEXT                                                    ┘
1091 FOR X=74 TO 0 STEP -1'                 ;MOVE LEFT       ┐
1092   GOSUB 1120                                            │ 左に動かす
1093 NEXT                                                    ┘
1100 GOTO 1070'                             ;LOOP            ← 無限ループ
1110 '
1120 'PRINT BEAM                            ;IN=X,Y          ┐
1130 LOCATE X,Y  :PRINT "  ▌  "                              │
1140 LOCATE X,Y+1:PRINT " ▀▀▀ "                              │ ビーム砲出力ルーチン
1150 LOCATE X,Y+2:PRINT " ▟▀▙ "                              │
1160 RETURN                                                  ┘
```

第136図　オールBASIC版の完成

子を見ることができます(**写真10**)。

さて，このビーム砲の動きを見ていてどう思いますか？

「遅い！」

「ビーム砲がゆがんで見える！」

と思われるでしょう。ビーム砲がゆがんで見えるのは，ひとえにBASICの遅さのなせるワザです。すなわち，ビーム砲が三行あるため各行の表示に時間のズレが生じるからです。そのためゆがんで見えるのです。

《写真10》ビーム砲が左右に移動

引数(パラメータ)

これを解決するには，

ビーム砲を表示するサブルーチン

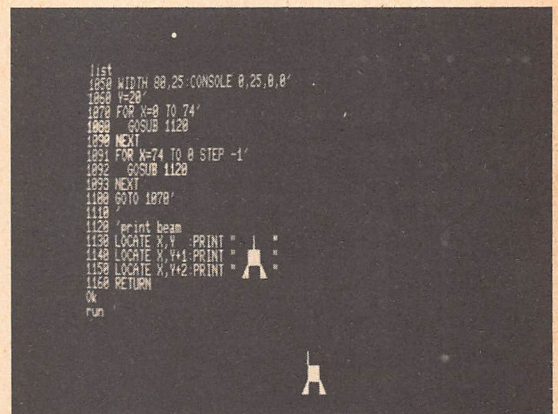
(第136図，1130行～1160行)

を**マシン語化**すれば良いのです。そこでUSR関数を用いることになります。しかし，先に我々が覚えた方法では，少し足りない面があるのです。というのは，メイン・ルーチンで指定した**ビーム砲の位置**

ヨコ：変数X

タテ：変数Y

の値を**マシン語のサブルーチンに渡してやらなければならない**からです。さもないと，マシン語側では

ビーム砲をどこに表示する？

のかわかりません。

したがってUSR関数を用いてマシン語サブルーチンを呼び出す際に，マシン語側に

引数（パラメータ）X，Y

の**値を渡してやる**必要があります。それには，先のUSR関数の使い方だけでは不十分であり，もう少し高級なUSR関数の使い方をマスターしなければなりません。

これを理解するのは，少々面倒かもしれません。そこでブロックを改め，次ブロックにおいてこの問題に挑戦して行くことに致しましょう。

〈リビング・ルーム〉

—USR関数とマニュアル—

一般にパソコン，マイコンを購入しますと，**マニュアル**がついてきます。そして，最近のマニュアルはどこのを見ても，かなり**丁寧**なものとなっています。とりわけ**BASIC 言語**の部分については，詳しく，親切であり，まるでＢＡＳＩＣの教則本の如くです。

しかし，それを読み進めていくと，最後のほんの一部分のところで，突然難しくなります。今まで丁寧であった説明が，突如として不親切になり，何やらムニャムニャと書いてあります。それは大抵，

ＵＳＲ関数
ＰＥＥＫ
ＰＯＫＥ

のように**マシン語に関する部分**であることが多いようです。

こういった事情は，昔の**キット式マイコン**のマニュアルにも見られたようです。マニュアルの最初の部分には，組立て法が説明してあり，それこそ**初心者**を対象に手取り足取り書かれています。こうしてキットが完成すると，続いてマニュアルはＯＳ（モニタ）やプログラムの説明に入ります。そしてここから突如として難しくなるのです。しかも対象が入門者から，一定水準以上の**技術者向け**となります。ちなみにキット式のマイコンの言語は，

マシン語だけ

でした。こうして有名な

キットを買った！
作った！　でも，動かなかった！

という現象（言葉）が生まれたのです。

ＰＣ－8001のマニュアルにも似たような現象があります。Ｎ－BASIC 言語のマニュアル部は，かなり丁寧です。しかし，ことＵＳＲ関数の部分になるとやや舌足らずのような感があります。しかし，しかしですよ——。

それにしてもメーカーは，マニュアルにおいて別にその部分を隠しているわけではありません。マニュアルを良く読むと，ちゃんと

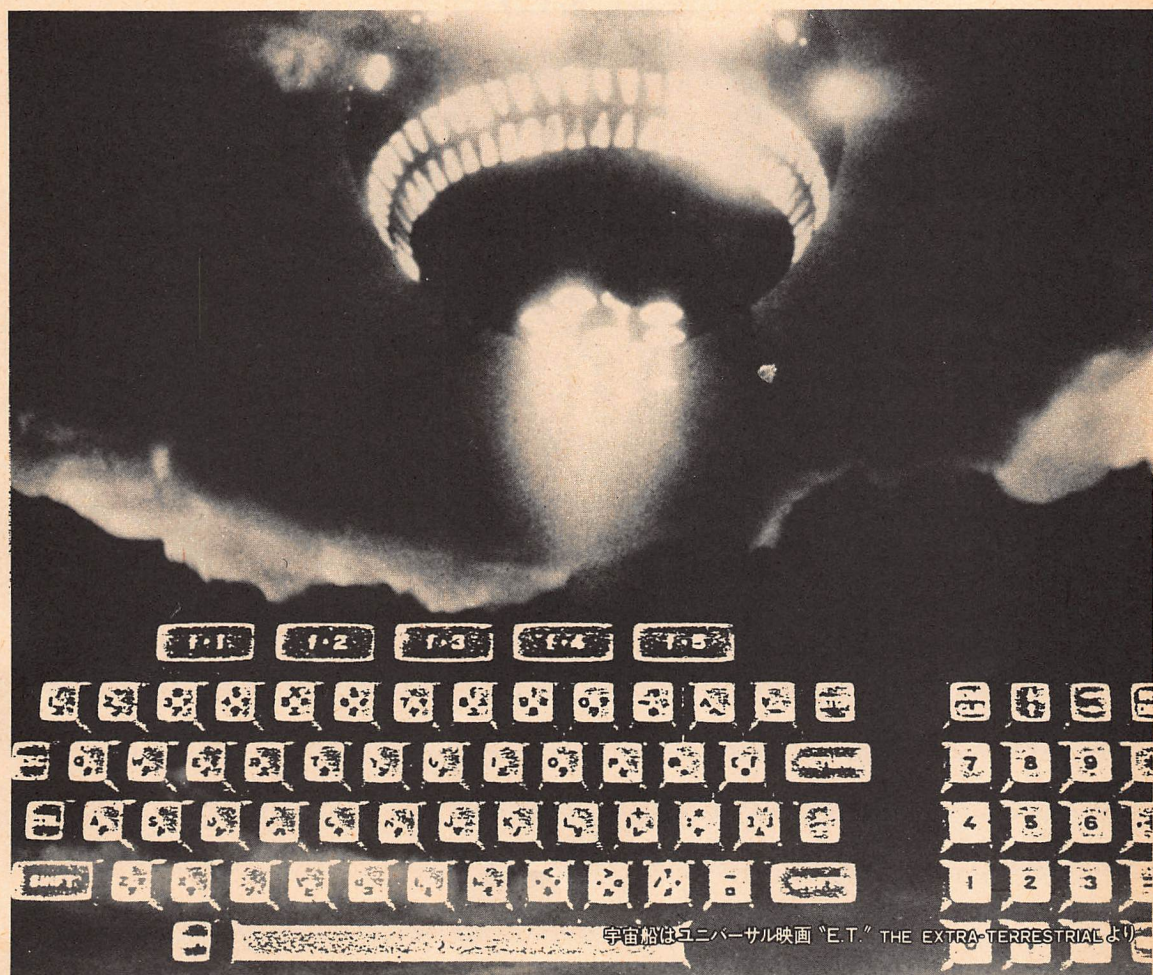
必要な情報は提供されている

のです。ただし，それにはマニュアルの隅から隅まで良く読み，若干の推察力が要求されます。次ブロックでは，そこらあたりにもメスを入れ，

ＵＳＲ関数の完全理解

を目差して頑張って行きましょう。

第3ブロック

浮動小数点型式とストリング・デスクリプタを探る

宇宙船はユニバーサル映画"E.T." THE EXTRA-TERRESTRIALより

〈はじめに〉

プログラミングの

生産性向上：高級言語を使う

速度の向上：低級言語を使う

という相反する目標を実現するため，一般に

BASIC＋マシン語

のリンクという手法が取られます。それには，

USR関数を使う

ことは，前ブロックでマスターしてきました。しかし，より一層あなたのＰＣを使いこなそうとする時，もう一歩つっこんで**ＵＳＲ関数のしくみ**を見ていく必要があります。すると，どうしても

浮動小数点アキュームレータ

ストリング・デスクリプタ

という**二つの壁**を突破していかなければなりません。そして，そこを足掛りとして，さらにあなたは前進していくことになるでしょう。

御健闘をお祈りします。

第10章

浮動小数点アキュームレータ

```
DD80: 10 F3 01 19 50 CD 3A 09 01 02 0C 11 9E E1 21 14 ; 0451
DD90: 14 CD 9C DD 06 04 CD 79 D9 10 FB C9 C5 E5 C5 D5 ; 099B
DDA0: CD 0C D9 D1 C1 1A 77 13 23 10 FA E1 C1 2C 0D 20 ; 0710
DDB0: EB C9 CD 13 DE CD CD DD 2E 08 3E E8 CD 60 D9 21 ; 096C
DDC0: 08 1A CD 0C D9 11 67 E1 CD E6 DD 18 2D 21 07 00 ; 062A
DDD0: CD 0C D9 06 05 E5 0E 78 36 00 23 0D 20 FA E1 11 ; 059A
DDE0: 78 00 19 10 F0 C9 1A A7 C8 77 13 23 23 FE 20 28 ; 05F9
DDF0: F5 D5 E5 CD 79 D9 E1 D1 18 EC 3A AB E3 A7 C8 2E ; 0AE9
DE00: 0A 3E A8 CD 60 D9 21 0A 10 CD 0C D9 11 73 E1 CD ; 0715
DE10: E6 DD C9 AF 32 AB E3 ED 5B C6 E3 2A C8 E3 CD D3 ; 0B61
DE20: 5E D0 ED 53 C8 E3 AF D3 51 11 FF 35 21 2D E0 CD ; 092C
DE30: 37 DA 01 19 50 CD 3A 09 3E FF 32 AB E3 C9 3A D1 ; 075C
DE40: E3 FE 03 D0 3A B1 E3 FE 0C 38 06 3A AB E3 E6 1F ; 0897
DE50: C0 CD 95 D9 E6 01 28 4F 3A AC E3 3C 67 3A BB E3 ; 089D
DE60: 6F 06 05 C5 E5 CD 6F DE E1 C1 2D 2D 10 F5 C9 E5 ; 08ED
DE70: CD 0C D9 7E 21 3D E1 01 16 00 ED B1 E1 C0 2C 2C ; 071D
DE80: DD 21 D2 E3 DD 7E 00 A7 28 06 DD 23 DD 23 18 F4 ; 07EF
DE90: DD 75 00 DD 74 01 DD 36 02 00 00 00 00 00 00 21 ; 03DA
DEA0: D1 E3 34 E1 E1 E1 C9 2A BB E3 24 24 3A BA E3 E6 ; 0A21
DEB0: 02 28 04 7C D6 37 67 06 0B CD 8B D9 A7 28 A2 F5 ; 06C6
DEC0: 7C C6 05 67 F1 3D 18 F4 DD 21 D2 E3 DD 7E 00 A7 ; 089D
DED0: C8 CD DA DE DD 23 DD 23 18 F2 DD 6E 00 DD 66 01 ; 08E6
DEE0: CD 0C D9 7E FE 5A 20 02 36 00 01 78 00 09 7E A7 ; 0587
DEF0: 20 0A 36 5A DD 7E 00 3C DD 77 00 C9 CD 3A DF DD ; 0731
```

ゆーざー・かんすう………

付 録 Ⓑ へ

> **USR**
> **書式**：USR ［〈数字〉］（X）
> **動作**：引数Xを持ってユーザーのアセンブリ言語ルーチンを呼び出します。〈数字〉は0から9までの整数で，DEFUSR文で定義した番号に対応します。付録Ⓑを参照してください。

これは，「PC－8001　N－BASICリファレンス・マニュアル」P. 87の**ユーザー関数**の部分です。〝〈数字〉は0から9までの整数で，DEFUSR文で定義した番号に対応します。〟の部分は，すでに前ブロックでやったとおりで，我々でも良くわかりますね？そして，

〝引数Xを持って……呼び出します。〟の部分が，どうやら，我々の狙いである

引数（パラメータ）の授受

のことをいっているようです。そして，その下に例が出ているのですが，これだけでは使い方が良くわかりませんね。そこで**藁をも掴む**思いで〝付録Ⓑを参照してください〟の御託宣にしたがって，

付　録　Ⓑ　(P. 97)

を見ることになります。

まず，

〝ユーザーのアセンブリ言語サブルーチンのためのメモリスペースは，それをロードする前に確保しておかなければなりません。〟

の部分でけつ躓くかもしれません。この部分は，重要ですから，ここらあたりから**解剖のメス**を入れていくことに致しましょう。

メモリ・マップ

まず，**メモリ・マップの使い方**です。

第137図を御覧ください。これが，PC－8001で使われているCPU，Z－80の**全アドレス空間**です。

0000番地～FFFF番地

が使えます。ただし，標準のＰＣ－8001がこのすべてのメモリを使えるわけではありません。

第138図が，ＰＣ－8001における**メモリ・マップ**（メモリの使用状況を示した図）です。

左側：32Ｋシステム
（増設ＲＡＭを取り付けたＰＣ－8001）
右側：16Ｋシステム
（買ったばかりのＰＣ－8001）

となっています。どちらのシステムでも

００００番地～５ＦＦＦ番地
ＲＯＭエリア＝システム部

となっていて，この部分に

ＢＡＳＩＣインタプリタ
マシン語モニタ

が入っています。

後半の

８０００～ＦＦＦＦ：**32Ｋシステム**
Ｃ０００～ＦＦＦＦ：**16Ｋシステム**

がいわゆる**ＲＡＭ領域**で，この部分にプログラムを書いたり，DATAを記憶させせることができるわけです。

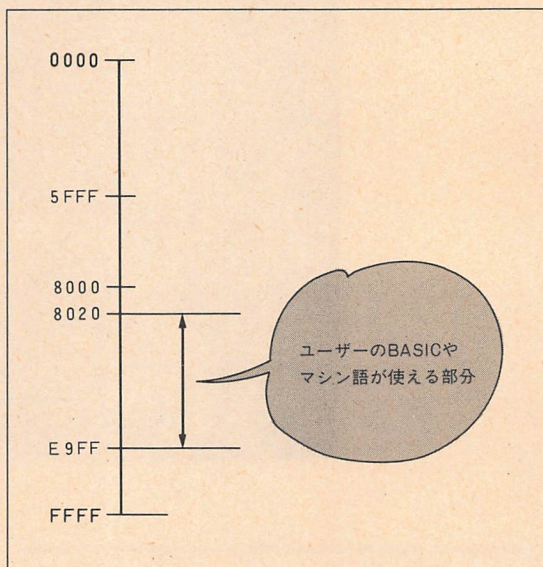
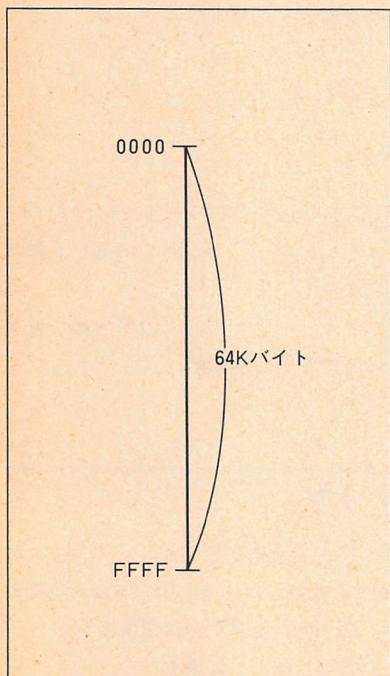
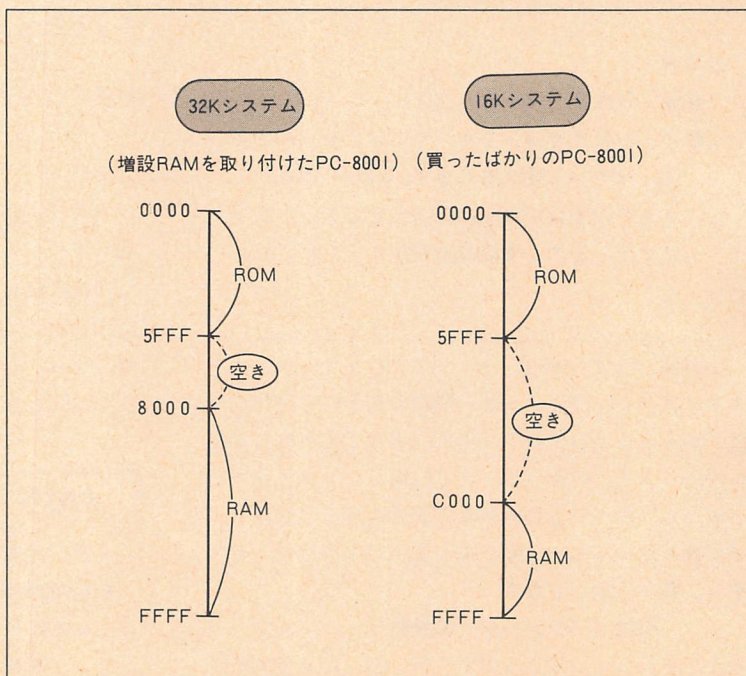
さらにこの**ＲＡＭの部分**を詳しく見てみます。ただし，以下の部分では32Ｋシステムの場合で説明していますので，16Ｋシステムの人は，

8──→C

のように置き換えてお読みください。

さて，ＲＡＭ領域ですが，この部分すべてをユーザー（利用者）が使えるわけではありません。**第139図**をご覧ください。図のように

８０２０番地～Ｅ９ＦＦ番地

の部分が，実際にユーザーの使える部分です。他の領域はシステムが使っています（空いている部分もありますが）ので，使わない方が安全でしょう。なお，以

第139図　ユーザーが使える部分

第137図　全アドレス空間

第138図　ＲＯＭエリアとＲＡＭエリア

下このユーザーの使える領域のことを

ユーザー領域

と略称することにします。

マシン語安住の地

さて，なぜこの時期にわざわざメモリ・マップの話しを持ち出して来たのでしょう？　もし**オール・マシン語**でプログラムを組むのであれば，これから述べることはあまり考える必要はありません。前節でみた

ユーザー領域（８０２０Ｈ～Ｅ９ＦＦＨ）

の中を自由に使ってください。しかし，ＵＳＲ関数を用い，

ＢＡＳＩＣ＋マシン語

でプログラムを組もうとすると，どうしてもユーザー領域の中身まで立ち入る必要があります。と申しますのは，

ＢＡＳＩＣ部……ＢＡＳＩＣのプログラム＋α
マシン語部…………マシン語のプログラム＋α

の両者がこのユーザー領域を使うことになるので，いい加減にプログラムを書くと，一方のプログラムで一方のプログラムが破壊されてしまう，ということが起きるからです。

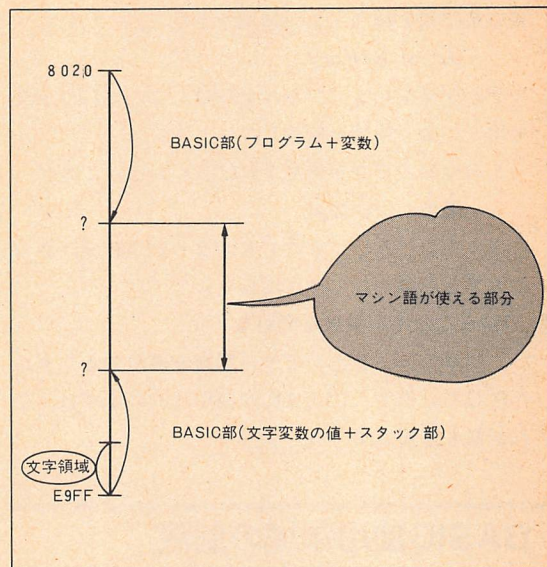
そこで**第140図**をご覧ください。これは，全メモリ・マップの中からユーザー領域を取り出したものです。図のようにBASICは，プログラムが大きくなるにしたがって

８０２０Ｈから後方へ
プログラム＋変数領域
Ｅ９ＦＦＨから前方へ
文字領域＋スタック領域

とメモリが使われて行きます。したがってマシン語は，真中の空いた部分を使うのが安全のようです。

しかし，——。

この空白地帯のアドレスは，何番地から何番地まででしょう。困ったことに，これがわからないのです。BASICのプログラムを実行したり，変更したりするたびにフラフラと変わっていきます。それも，前から後からと。

これでは危なくて，うかつにマシン語のプログラムを置くことはできません。

第140図　マシン語が使える部分は

CLEAR文

そこで**CLEAR**という命令が登場することになります。

> **CLEAR**
> **書式**：[[整数表記]，整数表記]
> **目的**：すべての変数を０に，文字変数をヌル・ストリング（長さ０の文字列）にクリアし，オプション指定により文字領域の大きさと使用メモリの上限を設定します。

ＣＬＥＡＲ文は，次の**三つの目的**のために使用します。

① **変数の値の初期化**

変数は御存知のように**数値変数**と**文字変数**があります。ＣＬＥＡＲ文を用いると，

各数値変数の値＝０
各文字変数の値＝""（ヌル・ストリング）

にクリアされます。

② **文字領域の大きさを決める**

プログラム中に文字変数を用いると，その文字変数の値（文字列）は，**文字領域**（**第140図参照**）に格納されます。通常，この文字領域の大きさは，

300バイト

用意されています。しかし，たくさんの文字変数や

長い文字列を使うとこの領域が不足し，

out of memory

エラーが発生します。そんな時は，このCLEAR文を用いて文字領域を増やしてやります。

どの位の大きさを用意すれば良いかは，まさに

適当，いい加減

に定めてください。それでもエラーが出れば，さらに大きさを増やしてやれば良いのです。

③ **BASICの使用領域を定める**

この目的は，まさに**USR関数を使用する**ために存在しています。CLEARの使用目的の中では，これが最も重要です。

BASIC使用領域の制限

そこで，例をあげて説明致します。仮に

CLEAR　300，&HC7FF↘

とキー・インしたとします。これは，**第141図**のように**プログラムの中**で使っても結構です。その時は，

RUN↘

すれば同じ結果が得られます。

まず最初のCLEARで，**変数の値がクリア**されます。続いて次の

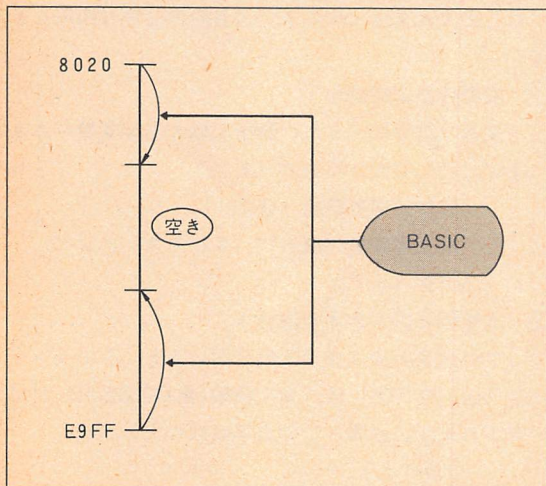
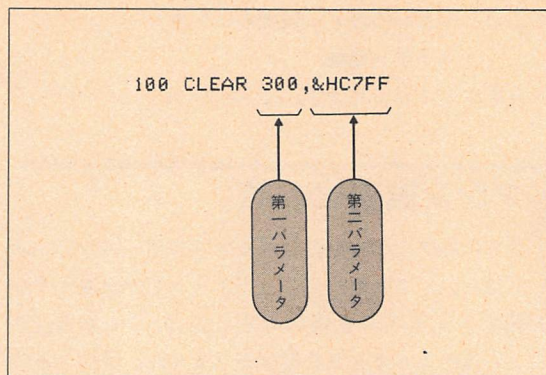
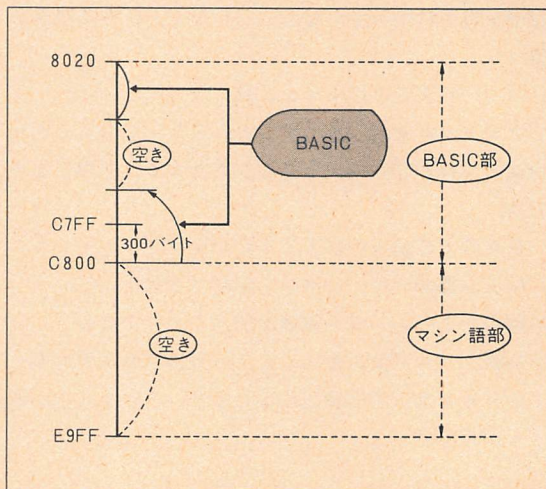
第1パラメータ＝300

で

文字領域＝300バイト

が指定されます。そして次が問題の

第2パラメータ＝&HC7FF

です。これにより**BASIC の使用領域の上限**を制限す

第142図　BASICの使用領域(初期状態)

第141図　CLEARをプログラム中で使う

第143図　CLEAR　300、&HC7FF宣言後

ることができます。

最初，**CLEAR**文を使う前のBASIC使用領域は，

8020H～E9FFH

でした(**第142図**)。しかし，**CLEAR文宣言の後は，**BASICは

8020H～C7FFH

しか使わなくなります(**第143図**)。したがいましてユーザー領域の後方

C800H～E9FFH

に空き領域ができます(ちなみに，C7FFHの次はC800Hですよ)。**この部分にマシン語を配置すれば**良いのです。

もちろん，前方BASIC使用領域の真中にも空き領域はできますが，こんな不確定要素の強い領域は使用しません。

<定　石>
　ＵＳＲ関数を用い，ＢＡＳＩＣとマシン語の両方を使用する時は，ＣＬＥＡＲ文を使って，ＢＡＳＩＣをユーザー領域の前方に追いやる。

引数の型

以上が，マニュアルP.97
〝ユーザーのアセンブリ言語サブルーチンのためのメモリ・スペースは，それをロードする前に確保しておかなければなりません。〃
の部分です。続いて，お目当ての

引数の引き渡し

の部分に入ります。マニュアル P．98の上の方を御覧ください。

ＵＳＲ関数の呼び出しが行なわれると，Ａレジスタはその関数に与えられた引き数の型を示す値を持ちます。その値は次のいずれかです。

Aの値	引き数の型
2	2バイトの整数（2の補数表示）
3	文字列
4	単精度浮動小数点数値
8	倍精度浮動小数点数値

ここも読んだだけでは，意味をつかみにくいかもしれません。実験を通じて理解しましょう。
まず〝引数の型〃の意味です。これは，

Ｘ＝ＵＳＲ（1）
　　　　↑
　この部分の定数（または変数）の型

のことです。ＵＳＲの（　）の中には，

整　数
単精度数値
倍精度数値
文　字

の**4種類**を用いることができます。そしてＵＳＲ関数が呼び出され，プログラムの制御がマシン語に移ったとたん，Ａレジスタの値は，その4種類のいずれかに応じて表のようにセットされるというのです。
以上を確かめるために，次のような実験を行ってみます。

整数型を用いると

まずＵＳＲ関数を用いるため，

マシン語サブルーチン

を用意します。アドレスは，

Ｃ８００Ｈ

から。そして，そのマシン語の中身はシンプルです。

ＦＦＨ

の1バイトだけ。これ，何だかわかります？　そうです。ＦＦＨとは，〝ミニ・レジスタ表示プログラム〃を**STOPさせるための指標**でした。したがいまして，マシン語サブルーチンに飛び込んだとたん，プログラムはSTOPしてしまいます。そしてその時の，

レジスタ類の値

が表示されます。

そこでそのサブルーチンをメモリに書き込みましょう。**Ｓコマンド**を使います。第144図のとおりです。

```
*SC800
C800 3D-FF 3D-
*
```

第144図　マシン語サブルーチン(1バイトのみ)を作る

次は，これを呼び出すための**メイン・ルーチン**です。まずBASICを上方に追いやります。

CLEAR　３００，＆ＨＣ７ＦＦ

これでマシン語は，Ｃ８００Ｈから使えます。次に**ユーザー関数の定義**です。

ＤＥＦ　ＵＳＲ＝＆ＨＣ８００

ユーザー関数は，1個しか使いませんので**ＵＳＲにつける数字**は省略しました。
これで準備はできました。ＵＳＲ関数を呼び出しますよ。**代入文を実行**すれば良いですね。

Ｘ＝ＵＳＲ（1）
　　　↑
型に注目！

（　）の中に御注目ください。1となっています。これは，

整数型の定数

です。したがってマニュアルによると，

Ａレジスタの値＝2

```
100 CLEAR 300,&HC7FF
110 DEF USR=&HC800:X=USR(1)'                         ;セイスウ
120 END
```

第145図　メイン・ルーチンを作る

になるはずです。

でき上がったプログラムは，第145図です。さっそくこのプログラムを走らせてみましょう。

RUN↘

結果は，第146図のとおりです。USR関数が呼び出され，プログラムの制御がC800Hに飛んだとたんにSTOPしています。それは，

PC＝C800H

を見ればわかります。そして，Aレジスタの値に注目してください。

A＝02H

となっています。すなわち，

USRの引数に整数型を用いると，

Aレジスタの値＝02H

になることが確認されました。

各型で実験する

他の型についても調べてみます。まず**単精度数値**。

コントロールBでBASICに戻り，

LIST↘

とキーインし，先程のメイン・ルーチンを表示します。そして，USRの**(　)の中を単精度の数値に変更**します。単精度にするには，数のあとに！をつければ得られます（定数や変数の型については，ここでは説明致しません。詳しくは，「リファレンス・マニュアル」P.11を御覧ください）。そこで1のあとに！をつけ，

X＝USR（1！）

と変更してやります。

これで**USR関数の引数**が，**単精度数値**になりました。でき上がったプログラムが，第147図です。

RUN↘

してみましょう（第148図）。**Aレジスタの値が変化したこと**に注目してください。

A＝04H

となっています。マニュアルどおりですね？

続いて**倍精度**の実験です。

コントロールBでBASICに戻りましょう。1のうしろの！を，**倍精度を表わす**＃に変えます(第149図)。

RUN↘

します(第150図)。やはりAレジスタの値は，変化しました。

A＝08H

となっています。やはりマニュアルどおりですね？

最後は，**文字型**の場合です。

やはりコントロールBでBASICに戻り，

1＃──→〝A〞

に変えます（別に〝A〞に限らず，文字型であれば何

```
¥SC800
C800 3D-FF 3D-
¥
Ok
LIST
100 CLEAR 300,&HC7FF
110 DEF USR=&HC800:X=USR(1)'                    ;セイスウ
120 END
Ok
RUN

AF   BC   DE   HL   IX   IY   PC   SP
026A C800 258A F0A8 9599 96CF C800 C6C1
¥
```


第146図　RUNでAレジスタを調べる

でも構いません)。変更されたメイン・ルーチンは，第151図のとおりです。

RUN↘

させますと，やはりAレジスタの値は変化します(第152図)。

A＝０３

となっています。やはりマニュアルのとおりです。マニュアルって，本当に正しいのですね？

FAC-3を求める

ＵＳＲ関数の引数について，以上まででわかったことをまとめておきましょう。

ＵＳＲ関数をCALLする時用いた引数［(　)の中の定数や変数］の型によって，**Ａレジスタの値が変化**する。逆にマシン語サブルーチン側では，Ａレジスタの値によって，**引数の型**を知ることができる。引数の型とＡレジスタの値の関係は，

引数の型	Ａレジスタ
整数型	０２Ｈ
単精度数値	０４Ｈ
倍精度数値	０８Ｈ
文字型	０３Ｈ

さて変数の型を，マシン語サブルーチン側から知ることはできるようになりました。しかし，これだけでは不足です。我々の知りたいのは，

メイン・ルーチンから
マシン語に
ある情報（ビーム砲の
位置等）
を伝達する方法

です。そこで更に「リファレンス・マニュアル」P.98 以下を読み進めていく事になります。

第147図　単精度に変えて

```
LIST
100 CLEAR 300,&HC7FF
110 DEF USR=&HC800:X=USR(1!)'                    ;ﾀﾝｾｲﾄﾞ
120 END
OK
RUN

AF   BC   DE   HL   IX   IY   PC   SP
046A C800 258A F0A8 9599 96CF C800 C6C1
*
```

単精度では

第148図　Ａレジスタ＝０４Ｈ

```
100 CLEAR 300,&HC7FF
110 DEF USR=&HC800:X=USR(1#)'                    ;ﾊﾞｲｾｲﾄﾞ
120 END
```

倍精度

第149図　倍精度では？

```
LIST
100 CLEAR 300,&HC7FF
110 DEF USR=&HC800:X=USR(1#)'                    ;ﾊﾞｲｾｲﾄﾞ
120 END
OK
RUN

AF   BC   DE   HL   IX   IY   PC   SP
086A C800 258A F0A8 9599 96CF C800 C6C1
*
```

倍精度では

第150図　Ａレジスタ＝０８Ｈ

第151図　文字型では？

マニュアルを読んでいますと，

数値の場合

文字の場合

で**処理の仕方が異なる**ようです。まず，**数値の場合**。〝引き数が数値の場合には，［HL］レジスタペアには引き数が格納されている浮動小数点アキュームレータへのポインタ（FAC－3）のアドレスが格納されています。〟

この部分でわかることは，引数に数値型を用いた場合，

HLレジスタ＝

（FAC－3）のアドレス

になるということです。(FAC－3)の意味がわからないかもしれませんが，とにもかくにもその**アドレス**を求めてみましょう。

```
LIST
100 CLEAR 300,&HC7FF
110 DEF USR=&HC800:X=USR("A")'                    ;ﾓｼﾞ
120 END
OK
RUN

AF   BC   DE   HL   IX   IY   PC   SP
036A 8001 EF58 F0A8 9599 96CF C800 C6C1
*
```

文字型では

第152図　Aレジスタ＝03H

```
LIST
100 CLEAR 300,&HC7FF
110 DEF USR=&HC800:X=USR(1)'                    ;ｾｲｽｳ
120 END
OK
```

整数型

第153図　整数型で（FAC－3）のアドレスを求める

USRの(　)の中に書く引数は，数値型であれば何でも良いのですから，先の整数型で実験してみましょう。プログラムは，**第153図**のとおりです。そして，

RUN↘

してみます。**第154図**のとおりです。ここまでは，先程の実験と同じですね？　ここでは**HLレジスタの値**に注目します。

HL＝F0A8H

となっています。これが，求める（FAC－3）のアドレスです。

浮動小数点アキュームレータ

（FAC－3）のアドレスが求まりましたので，次がいよいよメイン・ルーチンとマシン語サブルーチン間の情報の伝達の部分に入ります。

もう一度，「リファレンス・マニュアル」のいまの部分を読みますと，HLレジスタが

浮動小数点アキュームレータ

のポインタであると書かれています。マニュアルのこの部分を理解するには，どうしても〝**浮動小数点アキュームレータ**〟の意味を知らなくてはなりません。そこで以下を読まれる際の予備知識として〝浮動小数点アキュームレータ〟について，説明しておきます。

御存知のようにPC－8001のCPUは，Z－80です。したがって演算は，**基本命令**だけでは

16ビットの加減算

しかできません。しかし，N－BASICでは

倍精度の浮動小数点演算

第154図　RUN↘

を可能にしています。つまり基本的には不可能なことをやっているのです。それには，**何らかの仕掛け**があるのです。その仕掛けの基になっているのが，

浮動小数点アキュームレータ

です。アキュームレータとは，累算器のことでＪＩＳによる定義は次のとおりです。

> **累算器（accumulator）**
> 　レジスタの一種であって，その中に演算の結果が形成されるもの。

通常，演算は**アキュームレータ**を使って行います。しかし，Ｚ−80のアキュームレータ（Ａレジスタ）では，浮動小数点計算はできません。そこで，Ｎ−ＢＡＳＩＣでは，メモリ上に

8バイトの計算用領域

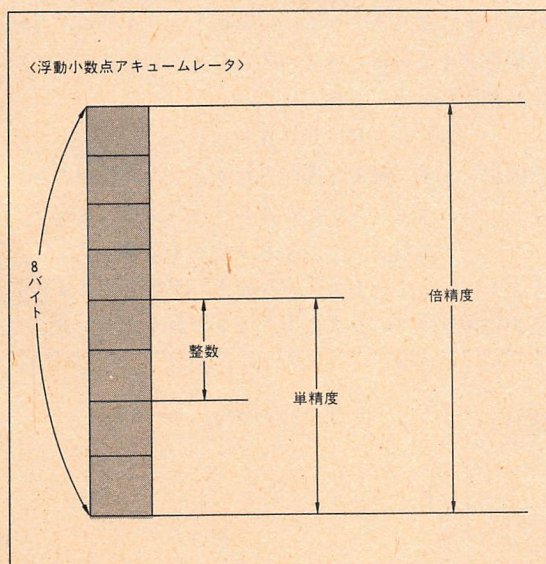
を作り，そのメモリ上で浮動小数点の演算を行っています。この8バイトの領域のことをマニュアルでは，

浮動小数点アキュームレータ

と呼んでいるのです。

引数に情報を乗せて

浮動小数点アキュームレータのしくみがどうなっているかは，「ＰＣ−8001　ユーザーズ・マニュアル」P.58で説明されています。

第155図を御覧ください。**倍精度**では，

第155図　浮動小数点アキュームレータ

7バイト──→仮数部
1バイト──→指数部

のように使います。**単精度**では，浮動小数点アキュームレータの後半4バイトしか使用せず，

3バイト──→仮数部
1バイト──→指数部

のようになっています。また**整数**では，さらにその半分の2バイトしか使用しません。

ここで再び「リファレンス・マニュアル」のP.98を御覧ください。

〝引き数が整数の場合には

ＦＡＣ−3が引き数の下位8ビットを，

ＦＡＣ−2が引き数の上位8ビットを保持します〞。

と書かれています。ここで結論を申し上げます。

> 引数が**数値型**の場合，ＵＳＲ関数がCALLされると，その**引数は浮動小数点アキュームレータに格納される。**

ということです。メイン・ルーチンから渡したい情報は，**ＵＳＲの（　）の中**に引数として書きます。マシン語サブルーチン側は，浮動小数点アキュームレータを覗けば，その情報を知ることができるというわけです。

（FAC−3）と（FAC−2）

それでは，具体的な情報の伝達法を説明していきましょう。

数値型の引数には，

整　数
単精度浮動小数点
倍精度浮動小数点

の三種類があります。そしてそれぞれの型に応じて，浮動小数点アキュームレータに**格納される方法**が異なります。その方法を理解しなければ，これらを利用することはできないわけです。そこで，まず最も簡単なモデルである整数型について調べてみます。しかし，有難いことにこの整数型が最も利用度が高いのです。

さて，その整数型の場合ですが，**第153図**のように

Ｘ＝ＵＳＲ（1）

を実行した時，この**引数の1**がどのような形で浮動小数点アキュームレータに格納されるか調べてみます。

整数型の場合，**2バイトの領域**に格納されることは，先に見たとおりです。そこで，引数の1を**2バイトの16進数に変換**してみます。

1＝0001H

ですね。ここで，

0001H

上位バイト　下位バイト

であることに注意してください。「リファレンス・マニュアル」によると，この

下位8ビット（01H）──→(FAC－3)

上位8ビット（00H）──→(FAC－2)

のように格納されると書かれています。先に調べましたように，

〔FAC－3〕＝F0A8H

です。したがいまして，

F0A8H＝01H

F0A9H＝00H

のように格納されるであろう，という結論に達します。

整数型のしくみ

これを実際に確かめるには，**第153図**，**第154図**に続いて，

F0A8H，F0A9Hの2バイト

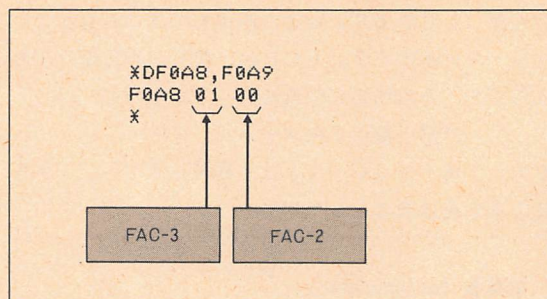
を調べてみれば良いのです。

DF0A8，F0A9↘

で調べます。結果は**第156図**のとおりです。ちゃんとマニュアルどおりになっています。

引数が，1ではあまりピンとこないかもしれません。もう一つ，例をあげてみます。いまメイン・ルーチンから，

1234H

というデータをマシン語に渡したいとします。もちろん，**1234Hだって整数型**ですよ。

第156図　FAC－3，FAC－2を確認する

$$1234H = 1\times16^3 + 2\times16^2 + 3\times16 + 4$$
$$= 4096 + 512 + 48 + 4$$
$$= 4660$$

これは，

－32768～32767

の間に入っていますから，立派な整数です。

さて，この1234Hをマシン語に送るわけです。それには，**USRの(　)の中に入れて代入文を実行すれば良い**のです。

X＝USR（&1234）

でOKです。したがってメイン・ルーチンは，**第157図**のようになります。前と同様に

C8FFH＝FFH

を書き込んでおき，〝ミニ・レジスタ表示プログラム〞をセットしておきます。そして，

RUN↘

を実行します（**第158図**）。これで**USR関数が実行されたこと**になりますから，引数の1234Hが，浮動小数点アキュームレータ（FAC－3とFAC－2）に格納されたはずです。

DF0A8，F0A9↘

で確認します。**第159図**のとおりです。御覧のように，

FAC－3：F0A8H＝34H

FAC－2：F0A9H＝12H

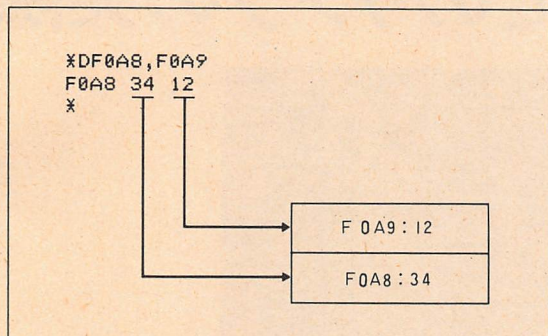
のようになっています。ちゃんと，引数1234Hの値が格納されていますね？　マシン語サブルーチンで

```
100 CLEAR 300,&HC7FF
110 DEF USR=&HC800:X=USR(&H1234)'                ;セイスウ
120 END
```


第157図　1234Hで調べる

```
RUN

AF   BC   DE   HL   IX   IY   PC   SP
026A C800 258A F0A8 9599 96CF C800 C6C1
¥
```

第158図　ＵＳＲ関数をＣＡＬＬする

第159図　浮動小数点アキュームレータを調べる

は，この値を取り出し，処理をすれば良いわけですね。

整数型引数のまとめ

以上が**整数型引数のＵＳＲ関数の使い方**です。まとめておきましょう。

《ＵＳＲ関数の使い方(整数型引数)》

① 　－32768～＋32767

の範囲にある整数（小数点は使えない）を**整数型の定数**という。

② 変数のあとに%をつけた変数，または

ＤＥＦ　ＩＮＴ　～

で定義された変数も**整数型の変数**という。

③ 整数型の定数，または整数型変数の値をマシン語サブルーチンに渡したい時には，ＵＳＲ関数の引数，すなわち(　)の中にその定数，または変数を用いる。

④ ＵＳＲ関数がＣＡＬＬされると，引数の値は

(ＦＡＣ－３)：Ｆ０Ａ８Ｈ＝下位バイト

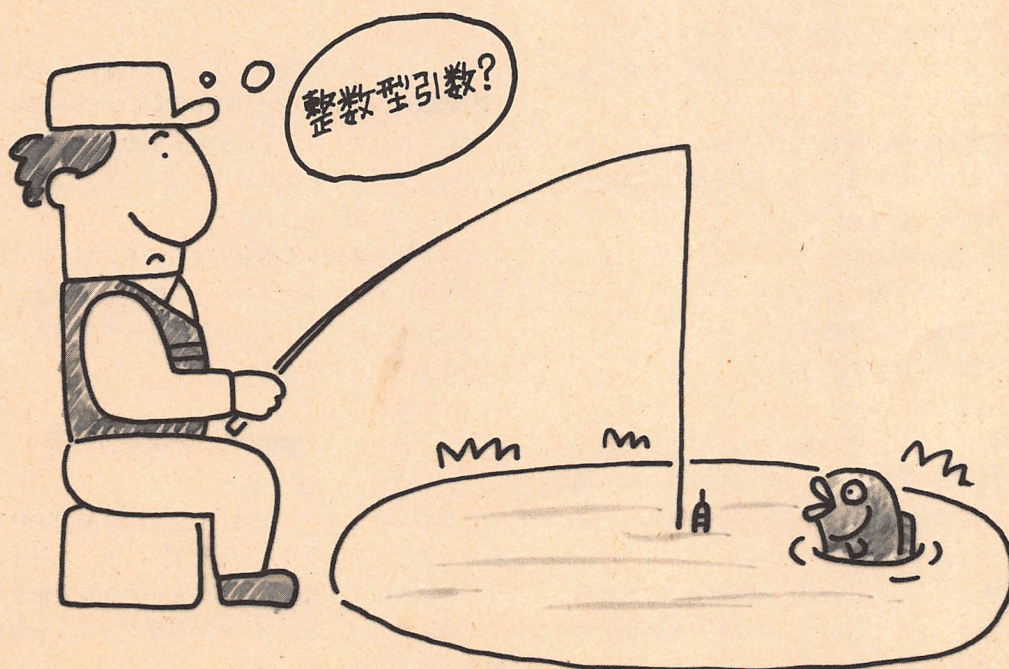
(ＦＡＣ－２)：Ｆ０Ａ９Ｈ＝上位バイト

のように格納される。

⑤ マシン語サブルーチンは，このアドレスを参照することで引数の値を知ることができる。

以上で本章は，幕を閉じることになります。そして，次章ではさっそくＵＳＲ関数を用いて，第136図

MOVING　BEAM　GUN

を，**マシン語化**してみることに致しましょう。

第11章

BASIC＋マシン語とのリンクの実際

```
DF00: 7E 00 FE 16 38 07 FE 18 30 03 CD 40 DF DD E5 FD ; 07C5
DF10: E1 FD 7E 02 A7 28 12 FD 7E 02 FD 77 00 FD 7E 03 ; 07AE
DF20: FD 77 01 FD 23 FD 23 18 E8 FD 36 00 00 FD 36 01 ; 071C
DF30: 00 21 D1 E3 35 DD 2B DD 2B C9 CD 3F DC E1 18 CD ; 0891
DF40: DD 6E 00 2C DD 66 01 CD 0C D9 7E 01 09 00 21 0C ; 0522
DF50: E0 ED B1 C0 CD 69 DF 3E 01 32 AA E3 21 D0 E3 35 ; 095A
DF60: C0 3E 03 32 AA E3 C3 53 D8 06 07 C5 11 53 E1 CD ; 0792
DF70: 87 DF 11 5D E1 CD 87 DF C1 10 F0 CD D1 D4 11 FF ; 0A2B
DF80: 17 21 70 E0 C3 37 DA 2E 17 3A AC E3 67 D5 CD 0C ; 077F
DF90: D9 D1 CD 1D D9 CD 95 D9 F6 07 47 0E 14 CD 0D DA ; 08C2
DFA0: 06 0A CD 73 D9 C9 0E 98 04 D8 02 58 00 80 48 EE ; 0684
DFB0: 84 08 00 90 35 33 53 09 00 80 48 EE 84 08 00 10 ; 0432
DFC0: B5 33 5B 01 00 C0 6C EE C6 0C 00 80 35 33 53 08 ; 0573
DFD0: 00 80 6C EE C6 08 00 10 B5 33 5B 01 00 E0 48 EE ; 0612
DFE0: 84 0E 00 80 35 33 53 08 00 80 48 EE 84 08 00 30 ; 0447
DFF0: B5 33 5B 03 00 01 03 08 09 10 30 33 35 53 5B 80 ; 0331
E000: 90 B5 B8 F8 F8 B8 00 B8 38 38 B8 00 0C 78 87 C0 ; 0850
E010: 00 7F 23 32 F7 00 A1 0C B8 4E 74 00 84 3F 11 43 ; 0509
E020: 84 00 10 87 AE 03 00 22 58 B6 13 1F 00 22 04 28 ; 037C
E030: 03 26 01 22 04 33 04 2D 01 28 01 26 01 22 01 26 ; 014E
E040: 02 28 02 2D 01 44 01 44 01 3D 01 44 01 4C 01 51 ; 0205
E050: 01 5B 01 28 01 26 01 22 01 1E 01 22 02 22 02 19 ; 0150
E060: 04 22 04 26 02 28 01 2D 01 2D 03 33 01 33 06 00 ; 0146
E070: 1E 04 1E 03 1E 01 1E 04 19 03 1B 01 1B 03 1E 01 ; 00F9
```

りんく・りんく………

ビーム砲のデータを用意して

前章は，やや理屈っぽい話しが続きました。かなりお疲れになったことと思います。本当に御苦労様でした。

USR関数の使い方，おわかりになったでしょうか？　理屈は二の次にするとしても，**使い方**だけはマスターしていただきたいものです。

ところで，引数の型は，

整　数　型
単精度浮動小数点型
倍精度浮動小数点型
文　字　型

の**4種類**がありました。そして，それぞれの型に応じてUSR関数の使い方は異なります。しかしながら前節では，まだ整数型の説明しか終っていませんでした。

しかし，――。

整数型の使い方さえわかれば，**90%以上**はUSR関数を使いこなすことができます。そこで，お約束どおり第136図のプログラムを実際に

BASIC＋マシン語化

してみることに致しましょう。

もう一度，第136図のプログラムを再掲いたします(**第160図**)。これは，**オールBASIC**でビーム砲を左右に動かすものでした。これを最小限のマシン語でスピード・アップするとすれば，やはり

1130行～1160行

の**ビーム砲を表示する部分**ですね。そこで，この部分に相等するマシン語サブルーチンを作ってみましょう。

何はともあれ，まず**ビーム砲のデータ**を用意します。**DC命令**を使えば簡単です。

DC　'　▌　'
DC　'　■　'
DC　'　◢▀◣　'

のように三行に分けて定義します。こうしておけば，後で見てもわかりやすいアセンブル・リストを作れます。これらのデータを使って，ビーム砲を表示するわけです。あとで，このデータの格納アドレスを参照し易すくするため，ラベルを

```
1000 '==================================
1010 '  MOVING BEAM GUN
1020 '       1982.11.7:BY K. TSUKAGOSHI
1030 '==================================
1040 '
1050 WIDTH 80,25:CONSOLE 0,25,0,0'           ;TV MODE SET
1060 Y=20'                                   ;SET LOCATE-Y
1070 FOR X=0 TO 74'                          ;MOVE RIGHT
1080   GOSUB 1120
1090 NEXT
1091 FOR X=74 TO 0 STEP -1'                  ;MOVE LEFT
1092   GOSUB 1120
1093 NEXT
1100 GOTO 1070'                              ;LOOP
1110 '
1120 'PRINT BEAM                             ;IN=X,Y
1130 LOCATE X,Y  :PRINT "  ▌  "
1140 LOCATE X,Y+1:PRINT "  ▄  "
1150 LOCATE X,Y+2:PRINT " ▟▀▙ "
1160 RETURN
```

マシン語化する部分

第160図　オールBASIC版(再掲)

1行目のデータ：D1

2行目のデータ：D2

3行目のデータ：D3

のように付けておくことにします(**第161図**)。

```
                          ORG   0C800H
                     ;
C800 20209788 D1:         DC    '  ▌  '
C804 2020
C806 20208787 D2:         DC    '  ▄  '
C80A 2020
C80C 20E49494 D3:         DC    ' ▟▀▙ '
C810 E520
C812                      END
```

このアドレスは，仮のアドレスですから
まだ入力しないでください。

第161図　DATAを作る

ビーム砲一行分の処理

データを3行に分けたのは，

1行目のデータを表示する
2行目のデータを表示する
3行目のデータを表示する

の**三部構成**でプログラム化しようと考えたからです。すると，同じようなことを三回繰り返すことになります。だったら，これら三行に共通する部分を**サブルーチン化**すると効率的ですね。仮にそのサブルーチンを

PLINE

と名付けますと，

D1のデータを指定
　CALL　PLINE
D2のデータを指定
　CALL　PLINE
D3のデータを指定
　CALL　PLINE

でプログラムを完成できます。

そこでPLINEの内容を考えてみましょう。まず**レジスタ・ペアの役割**を決定します。

DEレジスタ←―データのアドレス
HLレジスタ←―ビデオRAMのアドレス

のように割り振ってみます。すると，**1キャラクタの表示**は，

LD　A,　(DE)　―――①
LD　(HL),　A　―――②

で実現できます。簡単に説明致しましょう。**第162図**を御覧ください。本当は，

LD　(HL),　(DE)

とやりたいところですが，こういう命令はありません。そこで，①でまずキャラクタのデータをAレジスタに取り込みます。そして，②でそのデータをHLレジスタの指すビデオRAMに転送してやります。この二つの操作で，1キャラクタ分が表示されます。

続いて次のデータを表示するため

INC　DE　―――③

第162図　1キャラクタの表示

INC　HL ────④

を実行します。③でDEレジスタが，次のデータを指します。また④でHLレジスタが，ビデオRAMの右隣を指します。こうして，また①，②を繰り返せば良いのです。以上をまとめますと，

```
LD   A, (DE)     ┐
LD   (HL), A     │
INC  DE          ├──Ⓐ
INC  HL          ┘
```

このⒶの操作を6回繰り返せば，

ビーム砲1行分の処理

ができるわけです。

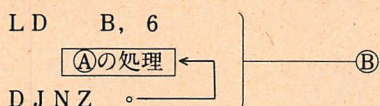
PLINEの完成

それには，Bレジスタにループ回数を入れ，

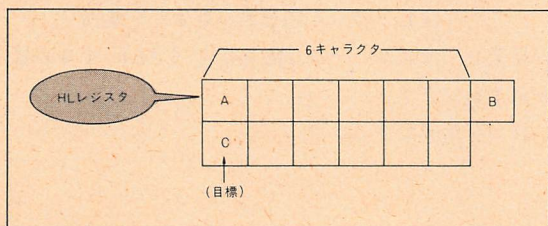
DJNZ命令

```
                  ;=================
                  ;  PRINT BEAM
                  ;      (82.11.18)
                  ;=================
                  ;
                         ORG   0C800H
                  ;
C800 0606                LD    B,6          ;6 CHARACTERS
C802 1A           PL1:   LD    A,(DE)
C803 77                  LD    (HL),A
C804 13                  INC   DE
C805 23                  INC   HL
C806 10FA                DJNZ  PL1
                  ;
C808 20209788 D1:        DC    '  ▮  '
C80C 2020
C80E 20208787 D2:        DC    '  ■  '
C812 2020
C814 20E49494 D3:        DC    ' ▟▀▙ '
C818 E520
                  ;
                         END
```


第163図　6回ループで1行

を使えば良いですね？　すると，この構成は

のようになります。そして，このⒷをプログラムで示したのが，**第163図**です。

以上が，サブルーチン **PLINE** の大きな役割りです。しかし，PLINE にはもう一つの任務を果さすべきです。それは PLINE が終了した後，さらにビーム砲の次の行を表示しなければならないからです。それには，**HLレジスタの値**をビデオＲＡＭ**次の行のビーム砲の先頭**を指すようにしておくと便利です。

第164図を御覧ください。最初ＨＬレジスタは，**A点**を指していました。PLINE 終了後は，６キャラクタ分右に移動し，**B点**を指すことになります。それを更に移動させ，**C点**を指すようにしてやりたいわけです。

第164図　ＨＬレジスタの移動

これのもっとも簡単な考え方は，**ビデオＲＡＭ一行がアトリビュート・エリアを含めて120バイト**あることを利用するのです。それには処理Ⓑの前後でＨＬレジスタの位置を変えないようにしてしまえば良いのです。すなわち，ＰＵＳＨ，ＰＯＰを使い，

```
PUSH  HL
   処理Ⓑ
POP   HL
```

としてやれば，処理ⒷのあとでもＨＬレジスタは**A点**を指します。しかる後に，ＨＬレジスタに**120**を足してやれば，**C点**を指すことになります。

```
LD    BC，120
ADD   HL，BC
```

でＯＫですね？　できあがった PLINE のプログラムは，**第165図**のようになります。

BEAMと名付けて

以上のように **PLINE の機能**は，

{ ＤＥレジスタ＝データのポインタ
　ＨＬレジスタ＝ビデオＲＡＭのポインタ

を**入力条件**として，

① ビーム砲の一行分のデータを表示する。

```
                      ;=================
                      ;  PRINT BEAM
                      ;      (82.11.18)
                      ;=================
                      ;
                               ORG   0C800H
                      ;
C800 E5               PLINE:   PUSH  HL        ;IN(DE=DATA,HL=ADDRESS)
C801 0606                      LD    B,6
C803 1A               PL1:     LD    A,(DE)
C804 77                        LD    (HL),A
C805 13                        INC   DE
C806 23                        INC   HL
C807 10FA                      DJNZ  PL1
C809 E1                        POP   HL
C80A 017800                    LD    BC,120    ;HL=HL+120
C80D 09                        ADD   HL,BC
C80E C9                        RET
                      ;
C80F 20209788 D1:              DC    '  |  '
C813 2020
C815 20208787 D2:              DC    '  ■  '
C819 2020
C81B 20E49494 D3:              DC    ' ▟▀▙ '
C81F E520
                      ;
                               END
```


第165図　PLINEの完成

② ＨＬレジスタの値を一段下げる。
の二機能を受け持つものでした。

さて，マシン語サブルーチン全体に

BEAM

というラベルを付けることにします。すでに PLINE が完成していますから，それを利用しますと **BEAM** 全体の構成は，

```
LD    DE, D1  ┐
              │（1行目の表示）
CALL  PLINE   ┘
LD    DE, D2  ┐
              │（2行目の表示）
CALL  PLINE   ┘
LD    DE, D3  ┐
              │（3行目の表示）
CALL  PLINE   ┘
```

となります。ところが3行目については，その真下に PLINE をつなげてやることにより，自動的に CALL したことになります。したがって，

```
BEAM：LD    DE, D1
      CALL  PLINE
      LD    DE, D2
      CALL  PLINE
      LD    DE, D3
PLINE：
        ⟆
      RET
```

というプログラムができ上がります。これをアセンブル・リストで示したのが，第166図です。

まずマシン語仮メイン・ルーチンで

まず，ここまでを実験してみましょう。とりあえず**オール・マシン語**で動くかを確かめ，BASIC のプログラムとリンクさせるのは，それからです。

さて，第 166 図はサブルーチンですからそのままでは走りません。メイン・ルーチンが必要です。そして
メイン・ルーチンの中で

ＨＬ＝ビーム砲を表示させたい場所

を指定し，サブルーチン **BEAM** を CALL する必要があるわけです。

話しを簡単にするため，

```
                    ;===================
                    ;  PRINT BEAM
                    ;      (82.11.18)
                    ;===================
                    ;
                            ORG   0C800H
                    ;
C800 111EC8         BEAM:   LD    DE,D1         ;DATA-1
C803 CD0FC8                 CALL  PLINE
C806 1124C8                 LD    DE,D2         ;DATA-2
C809 CD0FC8                 CALL  PLINE
C80C 112AC8                 LD    DE,D3         ;DATA-3
                    ;
C80F E5             PLINE:  PUSH  HL            ;IN(DE=DATA,HL=ADDRESS)
C810 0606                   LD    B,6
C812 1A             PL1:    LD    A,(DE)
C813 77                     LD    (HL),A
C814 13                     INC   DE
C815 23                     INC   HL
C816 10FA                   DJNZ  PL1
C818 E1                     POP   HL
C819 017800                 LD    BC,120        ;HL=HL+120
C81C 09                     ADD   HL,BC
C81D C9                     RET
                    ;
C81E 20209788 D1:           DC    '  ▌  '
C822 2020
C824 20208787 D2:           DC    '  ▀▀  '
C828 2020
C82A 20E49494 D3:           DC    ' ◢▀▀◣ '
C82E E520
                    ;
                            END
```

第166図　ＢＥＡＭのできあがり

ＨＬ＝Ｆ３００Ｈ

を指定してみましょう。これは，**ビデオＲＡＭの先頭アドレス**です。したがってビーム砲は，**ＴＶ画面の左上**に表示されることになります。

メイン・ルーチンは，

```
LD    HL, 0F300H
CALL  BEAM
JP    5C66H
```

で良いでしょう。このメイン・ルーチンを BEAM のあとに続けます。通常は，

メイン・ルーチン
↓サブ・ルーチン群

のようにプログラムが展開されます。しかし，ここではやがてメイン・ルーチンが削除されますから（あとで BASIC のメイン・ルーチンに置き換えられる），**メイン・ルーチンは付け足しの形**でサブルーチンのあとに置きました。

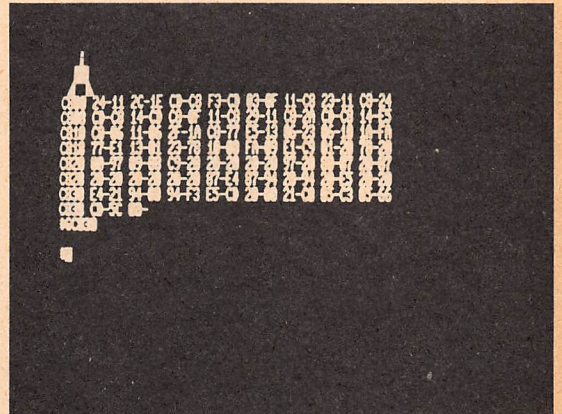
できあがったプログラムが，**第 167 図**です。さっそく走らせてみましょう。

ただし，注意です──。

プログラムの先頭アドレスは，

　　　Ｃ８３０Ｈ

です。ＯＫですね？　それでは，イザ。

　　　ＧＣ８３０↘

《写真11》左上にビーム砲表示

```
                        ;======================================
                        ;  PRINT BEAM (MAIN:MACHINE LANGUAGE)
                        ;       (82.11.18)
                        ;======================================
                        ;
                                ORG   0C800H
                        ;
03F3                    LOC:    EQU   3F3H            ;LOCATE TO ADDRESS
5C66                    MON:    EQU   5C66H
                        ;
C800 111EC8             BEAM:   LD    DE,D1           ;PRINT BEAM FROM MACHINE LANGUAGE
C803 CD0FC8                     CALL  PLINE
C806 1124C8                     LD    DE,D2
C809 CD0FC8                     CALL  PLINE
C80C 112AC8                     LD    DE,D3
                        ;
C80F E5                 PLINE:  PUSH  HL              ;IN(DE=DATA,HL=ADDRESS)
C810 0606                       LD    B,6
C812 1A                 PL1:    LD    A,(DE)
C813 77                         LD    (HL),A
C814 13                         INC   DE
C815 23                         INC   HL
C816 10FA                       DJNZ  PL1
C818 E1                         POP   HL
C819 017800                     LD    BC,120          ;HL=HL+120
C81C 09                         ADD   HL,BC
C81D C9                         RET
                        ;
C81E 20209788           D1:     DC    '  |  '
C822 2020
C824 20208787           D2:     DC    '  ■  '
C828 2020
C82A 20E49494           D3:     DC    ' ▟▀▙ '
C82E E520
                        ;
C830 2100F3             MAIN:   LD    HL,0F300H       ;V-RAM TOP
C833 CD00C8                     CALL  BEAM
C836 C3665C                     JP    MON
                        ;
                                END
```

（C830 への矢印注記：プログラムの先頭アドレス！）

第167図　マシン語仮メイン・ルーチンを付けて

結果は，**写真11**のとおりです。ビーム砲が，予定どおりＴＶ画面の左上に表示されました。サブルーチンＢＥＡＭは，正しく動いたようです。

レジスタの再定義

ここで第167図のプログラムで，一点程注意を申し上げておきます。先に分析しましたようにサブルーチンBEAMは，PLINEを三回CALLすることで成立しています。その際，PLINEを呼ぶ前に

ＬＤ　ＤＥ，［データの先頭］

をセットしています。しかし，本当はこれは最初の一回だけで良く，

Ｃ８０６Ｈ：ＬＤ　ＤＥ，Ｄ２
Ｃ８０ＣＨ：ＬＤ　ＤＥ，Ｄ３

の部分は不要です。これは，よくＤＥレジスタの動きを追っていけば，再セットが不要なことがわかります。しかし，そのように神経質にレジスタの値を追い，ほんの数バイトのことでプログラムがうまく動くかハラハラするより，第167図のようにズバッと**再定義**してしまった方がわかりやすいと思いませんか？　しかもその方がスッキリしていて，妙な**バグ**も減るようです。

現在の新しいプログラミングの立場では，第167図の書き方の方を勧めているようです。プログラムが長くなり，現在のレジスタの値がどのようになっているかわからなくなった時，どしどし

レジスタの値を再定義

することをお勧めします。健康のために。

LOCATE座標で指定する

さて，実験によりサブルーチン**BEAM**はどうやら正しく動くことが確認されました。しかしながら，もう少しこのサブルーチンを

加　工

しておく必要があるようです。と申しますのは，第167図の実験では，

ＬＤ　ＨＬ，０Ｆ３００Ｈ

の環境の元で行われました。これは，ビームの位置を

ビデオＲＡＭのアドレス

で指定しています。しかし，BASIC側から指定してくるビーム砲の位置は，

LOCATE座標

で指定しています。したがってサブルーチン BEAMでは，最初に

LOCATE座標
　　　　──→ビデオＲＡＭのアドレス

の変換を行う必要があります。それには，**システム**の中に便利なサブルーチンが入っていますから，それを利用することに致しましょう。それは，３Ｆ３番地から始まっています。仮にそのシステム・サブルーチンにＬＯＣという名前をつけることにします。

〈ＬＯＣ〉
番　地：３Ｆ３Ｈ
入力条件：
　Ｈレジスタ←──ヨコ座標
　Ｌレジスタ←──タテ座標
出　力：ＨＬレジスタ＝ビデオＲＡＭの番地
機　能：LOCATE座標を概当するビデオ RAMのアドレスに変換する。
　(注)　ヨコ座標，タテ座標は**１オリジン**で指定するものとする。

システム・サブルーチンLOC

それでは，具体的にシステム・サブルーチンＬＯＣの使い方を説明していきましょう。仮に

ＬＯＣＡＴＥ　２０，１０

の位置に相当する**ビデオＲＡＭの番地**を求めたいとします。このとき，

ヨコ座標＝２０　(10進数)
　　　　＝１４Ｈ (16進数)
タテ座標＝１０　(10進数)
　　　　＝０ＡＨ (16進数)

ですから，

Ｈレジスタ←──１４Ｈ
Ｌレジスタ←──０ＡＨ

のようにセットしてやります。すなわち

ＬＤ　ＨＬ，１４０ＡＨ

です。ここで重大な注意です。

ヨコ座標，タテ座標は１オリジンで指定

しなければならないということです。

１オリジン

というのは，**１から数えはじめる**という意味です。Ｌ

ＯＣＡＴＥは，**0オリジン**ですから０から数えます。したがって，テレビ画面左上の位置は，

（0，0）

です。しかし，ＬＯＣでＨＬレジスタに指定する座標は，左上が

（1，1）

です。すなわちLOCATEの数え方より**1大き**くしなければならないのです。

そこで，ＨＬレジスタにLOCATE座標を入れた後に，

ＩＮＣ　Ｈ　；ヨコ座標＋１
ＩＮＣ　Ｌ　；タテ座標＋１

してやらなければなりません。こうした後に

ＣＡＬＬ　ＬＯＣ

とすれば，**ＨＬレジスタにビデオＲＡＭのアドレスが求まります。**

LOCATE座標で実験

以上が，システム・サブルーチンＬＯＣの使い方です。さっそくこれを用いて実験してみましょう。

第167図のプログラムに修正を加えます。今度は，ＨＬレジスタにビデオＲＡＭのアドレスではなく，

LOCATE座標

でビーム砲の位置を指定してやります。仮に

LOCATE　5，5

の位置に表示させるものとします。するとメイン・ルーチンは，

ＭＡＩＮ：ＬＤ　　ＨＬ，０５０５Ｈ
　　　　　ＣＡＬＬ　ＢＥＡＭ
　　　　　ＪＰ　　ＭＯＮ

のようになります。サブルーチン BEAM の方も，最初に**座標変換の部分**を追加します。

ＢＥＡＭ：ＩＮＣ　　Ｈ

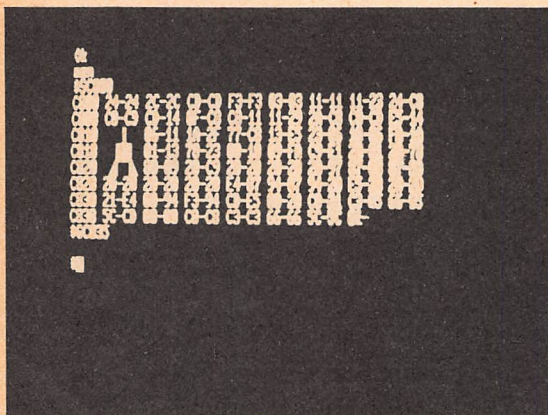
```
                    ;========================================
                    ;  PRINT BEAM (MAIN:MACHINE LANGUAGE 2)
                    ;        (82.11.18)
                    ;========================================
                    ;
                            ORG   0C800H
                    ;
03F3                LOC:    EQU   3F3H          ;LOCATE TO ADDRESS
5C66                MON:    EQU   5C66H
                    ;
C800 24             BEAM:   INC   H             ;PRINT BEAM FROM MACHINE LANGUAGE
C801 2C                     INC   L             ;(1,1) ORIGIN
C802 CDF303                 CALL  LOC
C805 1123C8                 LD    DE,D1
C808 CD14C8                 CALL  PLINE
C80B 1129C8                 LD    DE,D2
C80E CD14C8                 CALL  PLINE
C811 112FC8                 LD    DE,D3
                    ;
C814 E5             PLINE:  PUSH  HL            ;IN(DE=DATA,HL=ADDRESS)
C815 0606                   LD    B,6
C817 1A             PL1:    LD    A,(DE)
C818 77                     LD    (HL),A
C819 13                     INC   DE
C81A 23                     INC   HL
C81B 10FA                   DJNZ  PL1
C81D E1                     POP   HL
C81E 017800                 LD    BC,120        ;HL=HL+120
C821 09                     ADD   HL,BC
C822 C9                     RET
                    ;
C823 20209788 D1:           DC    '  ▎  '
C827 2020
C829 20208787 D2:           DC    '  ▀▀  '
C82D 2020
C82F 20E49494 D3:           DC    ' ▞▀▀▚ '
C833 E520
                    ;
C835 210505         MAIN:   LD    HL,0505H      ;LOCATE 5,5
C838 CD00C8                 CALL  BEAM
C83B C3665C                 JP    MON
                    ;
                            END
```

第168図　座標実験プログラム

```
INC   L
CALL  LOC
 ⌇
(以下同じ)
```

できあがりましたプログラムが，**第168図**です。プログラムを入力し，走らせてみましょう。先頭番地を間違えないように！

　　　ＧＣ８３５↘

でスタートです。結果は，**写真12**です。今回も予定通りの位置にビーム砲を表示させることができました。

《写真12》自由にビーム砲を表示できる

BASICとのリンク

さあ，いよいよ最後の仕上げ，

　　　BASICとマシン語のリンク

です。BASICのメイン・ルーチン，マシン語のサブルーチンの双方に手を加えます。

まず，メイン・ルーチン。

第160図のプログラムを御覧ください。

　　　1110行～1160行

は不要ですのでカットします。このビーム砲を表示する部分は，マシン語の方で用意するからです。

次に**ＵＳＲ関数**を使う準備をします。

　　　ＣＬＥＡＲ　３００，＆ＨＣ７ＦＦ

で，**BASIC部とマシン語の区分け**をしましょう。

　　　ＤＥＦ　ＵＳＲ＝＆ＨＣ８００

で**ＵＳＲ関数の定義**です。引数には，**整数型**を使用しますから，

　　　ＤＥＦＩＮＴ　Ａ－Ｚ

で**変数の整数定義**をします。これでＵＳＲ関数を使用する準備は，完了です。

最後の難関は，

　　　１１８０　ＧＯＳＵＢ　１１２０
　　　１０９２　ＧＯＳＵＢ　１１２０

をＵＳＲ関数に置き換える部分です。この部分は，少し難しいですよ。頑張って読んでくださいね。

ユーザー関数を使用するのですから，**代入文を実行**すれば良いですね？　代入される相手は，何でも構いません。ここでは，ＵＳＲのＵをとって

　　　Ｕ＝ＵＳＲ（?）

とすることにします。問題は，この(　)の中の引数をどうするか？――です。ここが，ヒジョ～～～にキビシイ所です。

2変数を1変数に

何がキビシイか？

いまメイン・ルーチンから渡したい情報は，**二つ**あります。

　　　ビーム砲のヨコ座標＝Ｘ
　　　ビーム砲のタテ座標＝Ｙ

の二つですね？　ところが，ＵＳＲ関数の引数は**一つ**しかありません。したがって，

　　　Ｘの値 ⎫
　　　　　　 ⎬──→一つの変数
　　　Ｙの値 ⎭

にまとめる必要があるのです。その一本にまとめた変数を，仮にＸＹとしましょう。するとＸＹは，ＸとＹに**適当な演算**をほどこして作り出すことになります。すなわち，

　　　ＸＹ＝Ｘ［ある演算］Ｘ

です。こうして一つの変数ＸＹが作り出されれば，

　　　Ｕ＝ＵＳＲ（ＸＹ）

でＵＳＲ関数を実行することができます。

変数ＸＹを作り出すには，条件があります。

① －32768～＋32767
　の**整数型の範囲**に収まること。

② ＸＹを２バイトの16進数に変換したとき，
　　　上位バイト＝Ｘの値
　　　下位バイト＝Ｙの値
　になること。

以上の二点です。

255進数の計算

この条件を満たしながら，変数X，Yから変数XYを作り出すことを考えてみます。

予備知識として，まず**10進数**の場合で考えてみましょう。たとえば，35という10進数は，

35＝3×10＋5

と分解することができます。逆に3と5という2数が与えられ，3を上位の数として一つの10進数を作れといわれた時には，

3×10＋5＝35

と計算して35という10進数を作り出すことができます。

X，YからXYという一つの変数を作り出すのも，まったく同様の考え方でできます。すなわち

Xという**1バイトの数**（上位）
Yという**1バイトの数**（下位）

⟹XYという**2バイトの数**

を作り出す，と考えれば良いのです。

ところで1バイトは，御存知のように

0～255(**256種類の数**)

までの数を表わすことができます。したがって，**バイトを単位**に数を数えると，

256進数

という数体系ができあがります。したがって，先程の10進数の考え方を用いると，

XY＝X＊256＋Y

という式でXYを作り出すことができるのです。

それでは，このことを紙とエンピツで確かめてみましょう。いま仮にX，Yの値が

X＝34
Y＝19

であったとします。これを先程の式に代入しますと，

XY＝X＊256＋Y
＝34×256＋19
＝8723

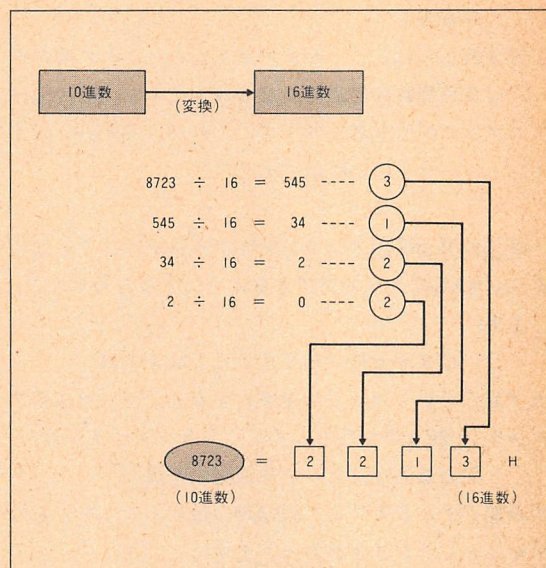
が得られます。8723を**16進数**に変換しますと，

$8723=2\times 16^3+2\times 16^2+1\times 16+3$

ですから，

8723＝2213H

となります(**第169図**)。したがって

XYの上位バイト＝22H

第169図　8723＝2213H

XYの下位バイト＝13H

です。これを**10進数に逆変換**すると，

上位バイト＝22H
＝2×16＋2
＝34（これはXの値と同じ）

下位バイト＝13H
＝1×16＋3
＝19（これはYの値と同じ）

となっていることがわかります。すなわち，

XY＝X＊256＋Y

で我々の希望どおりの変換が行えることが確認されました。

LD HL,(FAC-3)は不要

ちょっと難しかったかもしれませんが，前節の考察により，

ビーム砲を右に動かす部分

```
FOR X=0 TO 74
  XY=X*256+Y
  U=USR (XY)
NEXT
```

ビーム砲を左に動かす部分

```
FOR X=74 TO 0 STEP -1
  XY=X*256+Y
  U=USR (XY)
```

ＮＥＸＴ
と書き換えられることがわかりました。これに前にやった**ＵＳＲ関数の定義部分**をつけ加えると，**第170図**のプログラムができあがります。これが，BASICのメイン・ルーチンの完成版です。

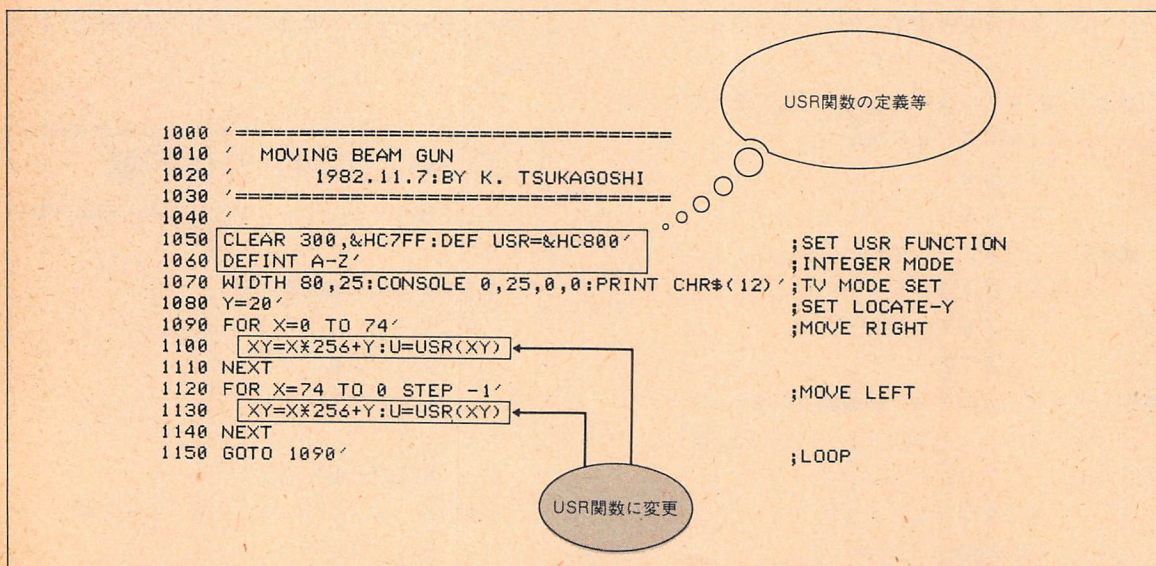
最後に第168図のマシン語サブルーチンに手を加えます。**BASICとリンクできるように**するためです。
まず，
　　Ｃ８３５Ｈ～Ｃ８３ＤＨ：ＭＡＩＮ
は不要ですからカットします。すると，マシン語サブルーチン**BEAM**が残ります。このサブルーチンは，
　　入力条件：ＨＬ＝ビーム砲の座標
が必要です。ですから，**BEAM**の最初のところでそれをセットしてやる必要があります。
〝ビーム砲の座標は，どこにあるか？〟
――（ＦＡＣ－３），（ＦＡＣ－２）に格納されています。
それなら，そこからそのデータ（情報）を取り出して，ＨＬレジスタに収めてやれば良いですね。

こんなことは，もうあなたにとっては朝飯前，ヨシノ屋のギュードンでしょう。まずＨＬレジスタに（ＦＡＣ－３）のアドレスをセットします――。
おっと，それは必要ないですね？　ＵＳＲ関数の引数に**数値**（文字以外）を用いると，自動的に
　　ＨＬ＝（ＦＡＣ－３）
になるのでした！

BASIC＋マシン語版が動いた

さあ，そこでまず一度ビーム砲の座標をＤＥレジスタに移しましょう。
　　ＬＤ　　Ｅ，（ＨＬ）
これで下位のバイト，すなわち**タテ座標**がＥレジスタに入りました。ここで
　　ＩＮＣ　Ｅ
とするのをお忘れなく！　システム・サブルーチンＬＯＣ（３Ｆ３Ｈ）を使用する時は，**１オリジン**で数えるのでしたね？
同様にして上位バイトを取り出します。
　　ＩＮＣ　ＨＬ
でＨＬレジスタは，（ＦＡＣ－２）を指しますから，
　　ＬＤ　　Ｄ，（ＨＬ）
　　ＩＮＣ　Ｄ
で取り出せます。これでビーム砲の座標が，**ＤＥレジスタ**に格納されましたから，
　　ＥＸ　　ＤＥ，ＨＬ
で**ＨＬレジスタ**に移してやります。
以上の手続きで，マシン語のサブルーチンが完成しました。**第171図**のとおりです。

さあ，長く頑張ってまいりました。
　　ＢＡＳＩＣ＋マシン語

```
1000 '==================================
1010 '  MOVING BEAM GUN
1020 '       1982.11.7:BY K. TSUKAGOSHI
1030 '==================================
1040 '
1050 CLEAR 300,&HC7FF:DEF USR=&HC800'              ;SET USR FUNCTION
1060 DEFINT A-Z'                                   ;INTEGER MODE
1070 WIDTH 80,25:CONSOLE 0,25,0,0:PRINT CHR$(12)'  ;TV MODE SET
1080 Y=20'                                         ;SET LOCATE-Y
1090 FOR X=0 TO 74'                                ;MOVE RIGHT
1100   XY=X*256+Y:U=USR(XY)
1110 NEXT
1120 FOR X=74 TO 0 STEP -1'                        ;MOVE LEFT
1130   XY=X*256+Y:U=USR(XY)
1140 NEXT
1150 GOTO 1090'                                    ;LOOP
```

第170図　メイン・ルーチン（BASIC）の完成

```
                     ;=================
                     ;  PRINT BEAM
                     ;      (82.11.18)
                     ;=================
                     ;
                            ORG   0C800H
                     ;
03F3                 LOC:   EQU   3F3H          ;LOCATE TO ADDRESS
                     ;
C800 5E              BEAM:  LD    E,(HL)        ;PRINT BEAM FROM BASIC
C801 1C                     INC   E             ;(1,1) ORIGIN
C802 23                     INC   HL
C803 56                     LD    D,(HL)        ;DE=LOCATE OF BEAM
C804 14                     INC   D
C805 EB                     EX    DE,HL
C806 CDF303                 CALL  LOC
C809 1127C8                 LD    DE,D1
C80C CD18C8                 CALL  PLINE
C80F 112DC8                 LD    DE,D2
C812 CD18C8                 CALL  PLINE
C815 1133C8                 LD    DE,D3
                     ;
C818 E5              PLINE: PUSH  HL            ;IN(DE=DATA,HL=ADDRESS)
C819 0606                   LD    B,6
C81B 1A              PL1:   LD    A,(DE)
C81C 77                     LD    (HL),A
C81D 13                     INC   DE
C81E 23                     INC   HL
C81F 10FA                   DJNZ  PL1
C821 E1                     POP   HL
C822 017800                 LD    BC,120        ;HL=HL+120
C825 09                     ADD   HL,BC
C826 C9                     RET
                     ;
C827 20209788 D1:           DC    '  |  '
C82B 2020
C82D 20208787 D2:           DC    '  ■  '
C831 2020
C833 20E49494 D3:           DC    ' ◢▔◣ '
C837 E520
                     ;
                            END
```

第171図　マシン語サブルーチンの完成

とのリンク，いよいよ最後の大詰めがやってきました。プログラムを走らせますよ。

第170図のメイン・ルーチンを入力してください。そして，今できあがったばかりの第171図，マシン語サブルーチンも入力します。もちろん

MON↘

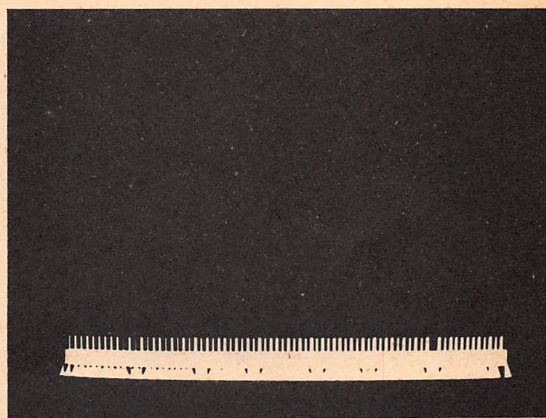
SC800

でお願いしますよ。

入力が終ったら，コントロールBでBASICのコマンド・レベルに戻ってください。そして，

RUN↘

でプログラム・スタートです。

いかがですか？　第160図のオールBASIC版とは，**スピードが比べものにならないでしょう？（写真13）**。

《写真13》マシン語＋BASICのビーム砲完成

第12章

浮動小数点型式とストリング・デスクリプタ

```
E080: 1E 03 20 01 1E 08 00 33 01 2D 01 28 01 22 02 28 ; 013F
E090: 01 22 04 00 00 28 0A A8 11 58 12 E8 1C A8 24 58 ; 03A4
E0A0: 25 E8 2F A8 34 58 35 E8 3D B8 3E 38 3F B8 40 E8 ; 0717
E0B0: 41 C8 45 28 87 87 87 87 87 87 87 87 87 87 20 20 ; 06FC
E0C0: 53 43 4F 52 45 09 39 39 39 39 39 20 20 20 20 20 ; 0342
E0D0: 48 49 2D 53 43 4F 52 45 09 39 39 39 39 39 20 20 ; 03A0
E0E0: 20 20 20 53 43 45 4E 45 09 39 39 39 20 20 20 20 ; 0302
E0F0: 20 4E 77 E4 3A EA EA EA 20 20 20 87 87 87 87 87 ; 07C4
E100: 87 87 87 87 87 C8 FE FF FF FF FF FF EF 8C FF FF ; 0CDD
E110: FF FF FF FF FF FF FF FF FF 7F 77 77 77 F7 FF FF ; 0DD0
E120: FF FF 00 00 00 00 00 FF FF 80 EC EE BF 99 FB EE ; 0997
E130: CE 08 00 33 E6 7F 22 77 22 F7 6E 33 00 01 03 08 ; 04CD
E140: 09 0C 0E 10 30 33 35 48 53 5B 6C 80 84 90 B5 C0 ; 0536
E150: C6 E0 EE 42 10 88 36 00 2C F6 F7 0C 00 79 C9 28 ; 0733
E160: 80 00 14 7B F3 E5 00 47 41 4D 45 20 4F 56 45 52 ; 055D
E170: 20 21 00 59 4F 55 20 47 4F 54 20 41 20 48 49 2D ; 0387
E180: 53 43 4F 52 45 20 21 21 00 33 01 2D 01 28 01 22 ; 028B
E190: 01 FF 02 33 01 2D 01 28 01 22 01 FF 02 00 2C 22 ; 02FF
E1A0: 04 00 00 2C 22 0C 00 00 00 0F 87 99 0F 00 00 87 ; 0223
E1B0: 88 07 00 00 00 0B 41 46 20 20 20 42 43 20 20 20 ; 0266
E1C0: 44 45 20 20 20 48 4C 20 20 20 49 58 20 20 20 49 ; 0327
E1D0: 59 20 20 20 50 43 20 20 20 53 50 02 01 48 1C C8 ; 037E
E1E0: 50 00 02 01 C8 50 00 48 20 49 20 54 20 20 20 4B ; 033B
E1F0: 20 45 20 59 20 20 20 21 20 21 20 20 20 20 20 20 ; 0260
```

ゆーざー・かんすうの　しくみ……

浮動小数点型式は後まわし

前章の実験でもおわかりのように，**USR関数**を導入し，

BASIC＋マシン語

のリンクにより，

開発の手間は最小に！

スピードは超高速に！

する手掛かりがつかめたのではないでしょうか？　前章の実験では，ビーム砲を表示する部分だけをマシン語化しました。それだけでも，これだけのスピードが出せるのですから。

ところで，我々はまだUSR関数の引数について学習の途中でした。引数の四つの型，

整　数　型
単精度浮動小数点型
倍精度浮動小数点型
文　字　型

のうち**整数型が終っただけ**です。そこで，残りの三つの型についても調べていくわけです。しかし，私はあえて次のことを提言します。

① 大部分の人は，ここから直ちに**次のブロック**に進んでください。
② どうしてもUSR関数に興味ある人は，ここから次の浮動小数点の部分はジャンプし，**文字型**のところから読み進めてください。
③ それでも次を読みたい人だけが，次の浮動小数点に関する部分をお読みください。

その理由は，次のとおりです。

① 整数型引数の使い方さえわかれば，**ほとんどの処理に困らない**こと。
② たまには引数に文字列を使いたいことも生じるので，文字型引数の使い方を覚えておいても**損はない**こと。
③ 浮動小数点型式については，マシン語で実数を扱わない限り，**ほとんど必要ない**こと。にもかかわらずそれをマスターしようとすると，かなり面倒であり，かつ若干の**予備知識**が要求されること。

等によります。すなわち必要のないことを苦労してマスターするより，とりあえず必要なことを先にやっておきましょう，という発想です。

単精度で考える

そこで，その**浮動小数点型式**の**引数の扱い方**に入ります。最初にお断りしておきますが，以下をお読みになるには，

2進数
16進数

の知識が必要です。しかも，

2進数 ｝ の小数点が扱えること
16進数 ｝

が大前提です。これらの予備知識については，コンピュータの入門書をお読みになりますと，いくらでも出てきますので，そちらで学んでください。さらには，

有効数字について

の知識がありますと，なおさら便利です。有効数字については，物理の本等で学んでください。

さて，何度も見てきましたように浮動小数点は，

単精度浮動小数点
倍精度浮動小数点

の二種類があります。しかし，その違いは格納すべき浮動小数点アキュームレータの**バイト数**だけです。ですから一方を理解すれば，他方は同様に扱うことができます。そこで，ここでは単精度についてのみ説明することに致します。

そこで例として，

5276.13

という単精度の数を用いることにします。これを引数として

X＝USR（5276.13）

のようにUSR関数を実行すると，どのように浮動小数点アキュームレータに格納されるかを考えてみましょう，というわけです。

有効数字による表現

浮動小数点は，浮動小数点内部では**2進数**に変換して処理されています。そこで，まず

5276.13 ⟶ ？
(10進数)　(16進数)

の変換を行ってみましょう。

小数点付きの10進数を2進数に変換するには，

＜ **整数部**：2で割り算
＜ **小数部**：2で掛け算

のように分けて行います。**第172図**のように計算しますと，

5276.13 (10進数)
＝1010010011100.00100001010 (16進数)

のように変換されることがわかります。

ここで10進数の場合で考えてみます。工学の分野で**有効数字**を明確に表現する場合，通常，

$n.nn\cdots n \times 10^{nn\cdots n}$

のように表現するのは，御存知だと思います。たとえば，

$724.54 = 7.2454 \times 10^2$

の如くです。

浮動小数点アキュームレータでもこの考え方を使います。すなわち，すべての2進数を

$1.mm\cdots m \times 2^{m\cdots m}$

のように表現します。2進数は0と1しかありませんから有効数字表現をすると，

1の位は必ず1

になります。

そこで先の

1010010011100.00100001010

を有効数字表現に変えますと，

$1.010010011100001000010 10 \times 2^{⑫}$
（注，これはまだ10進数です！）

のように表現されます（**第173図**）。

(注) 小数点付きの10進数を2進数に変換する際，うまく割り切れれば良いのですが，すべてが**割り切れる**とは限りません。したがってその際には，小数点下位の桁で**誤差**が生ずるのは避けられません。
この問題は重要で，実務等でBASICを利用する時には注意が必要です。最近ではBASIC内部でも浮動小数点を10進数で扱うものが増えてきていま

す。

仮数部の変換

さて，浮動小数点を

1．［ Ⓐ ］× 2［Ⓑ］

のような形で表現すると，**すべての数はⒶ，Ⓑの値のみで決定できる**ことに注意してください。浮動小数点アキュームレータでも，この**二つの数のみ**を記憶しています。そして

Ⓐの部分──→**仮数部**

Ⓑの部分──→**指数部**

と呼んでいます。

さあ，そこで，仮数部，指数部が浮動小数点アキュームレータにどのような形で格納されるかです。まずⒶの仮数部から。**第174図**を御覧ください。最初に小数部分を取り出します。

□01001001110000100001010
（小数部分）

次にこれに**符号ビット**を付け加えます。符号ビットは，

- **正　数**──→0
- **負**　──→1

です。5276.13は正数ですから，0を加えます。

0̇01001001110000100001010
↑
（符号ビットの追加）

できあがった2進数を16進数表現に変えます。

0010	0100	1110	0001	0000	1010
2	4	E	1	0	A

したがって，

24　E1　0A

という**3バイト**の数ができあがります。これを**逆順**にしますと，

0A　E1　24　────Ⓐ

これで仮数部の変換が終りました。

5276.13 ＝ 5276 ＋ 0.13

5276 ÷ 2 = 2638 …… 0	0.13 × 2 = 0.26
2638 ÷ 2 = 1319 …… 0	0.26 × 2 = 0.52
1319 ÷ 2 = 659 …… 1	0.52 × 2 = 1.04
659 ÷ 2 = 329 …… 1	0.04 × 2 = 0.08
329 ÷ 2 = 164 …… 1	0.08 × 2 = 0.16
164 ÷ 2 = 82 …… 0	0.16 × 2 = 0.32
82 ÷ 2 = 41 …… 0	0.32 × 2 = 0.64
41 ÷ 2 = 20 …… 1	0.64 × 2 = 1.28
20 ÷ 2 = 10 …… 0	0.28 × 2 = 0.56
10 ÷ 2 = 5 …… 0	0.56 × 2 = 1.12
5 ÷ 2 = 2 …… 1	0.12 × 2 = 0.24
2 ÷ 2 = 1 …… 0	⋮
1 ÷ 2 = 0 …… 1	

5276 ＝1010010011100（2進数）　　0.13 ＝0010000101 0…（2進数）

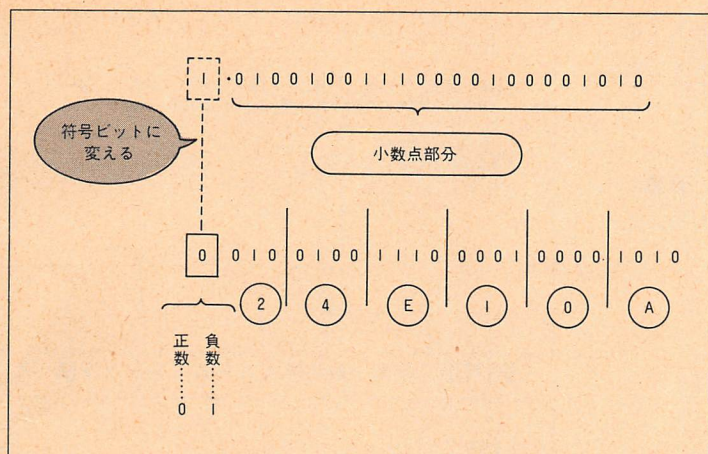
1010010011100.0010001010

第172図　小数点は10進数→2進数

第173図　有効数字で表現

指数部の変換

次に指数部の変換を見てみます。

2^{12}

第174図　仮数部の変換

ですから，**指数部の値**は

12＝**0 C H**

です。これが浮動小数点アキュームレータにどのような形で格納されるかです。

感覚的には，このまま０ＣＨが格納されるような感じがしますが，実際はこれに

＋81H

された値が格納されます。すなわち

0CH＋81H＝8DH

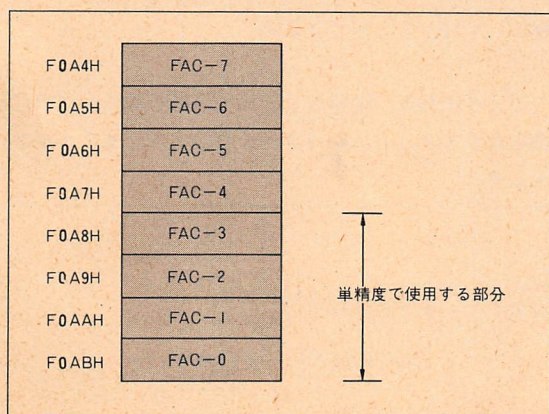
ですから，**8DH**が実際に格納される値です。

これを前節のⒶに続けると，

0A　E1　24　8D ———Ⓑ

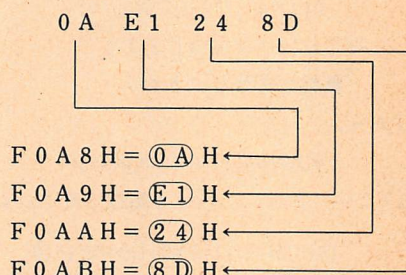
の**4バイトの数**が得られます。この4バイトが，単精度浮動小数点の値として浮動小数点アキュームレータに格納されます。

ここで第155図を御覧ください。同図のように

浮動小数点アキュームレータ＝8バイト

第175図　浮動小数点アキュームレータ

から構成されています。「リファレンス・マニュアル」では，これらの8バイトに**第175図**のように名前をつけています。そして単精度では，

(FAC−3)：F0A8H
(FAC−2)：F0A9H　**仮数部**
(FAC−1)：F0AAH
(FAC−0)：F0ABH；**指数部**

の4バイトを使っています。したがって先程のⒷの4バイトが，

に格納されるわけです。

負数の浮動小数点

以上を実験で確かめてみます。**〝ミニ・レジスタ表示プログラム〟**をロードし，F1E3Hからの3バイトの書き換えを行っておきます。**Sコマンド**で

C800H＝FFH

も書き込んでおきましょう。そして，メイン・ルーチンを用意します。メイン・ルーチンは，要するに

X＝USR（5276.13）

が実行できるようにあしらえば良いのです。**第176図**でOKです。

これら，入力等の準備が終りましたら，

RUN↘

です。レジスタの値が表示されてマシン語のコマンド・レベルになりますから，

DF0A8，F0AB↘

```
LIST
100 CLEAR 300,&HC7FF
110 DEF USR=&HC800:X=USR(5276.13)
120 END
Ok
```


第176図　メイン・ルーチンを作って

で（FAC—0）~（FAC—3）の値を調べます。結果は，第177図のとおりです。我々の変換どおりの結果が出ました。

いまのは，正数での実験でした。念のため，**負数**でも実験しておきましょう。数値は手間を省くため，先程用いた5276.13の正負を逆転し，

−5276.13

で実験してみます。

仮数部の小数点部分は，先程と同じですから

□01001001110000100001010

これに符号ビットを追加しますと，

負数：符号ビット＝1

ですから，

1̲01001001110000100001010

となります。さらにこれを**16進数**に変換しますと，

1010	0100	1110	0001	0000	1010
A	4	E	1	0	A

となり，できあがった**3バイト**が，

A4　E1　0A

です。逆順にして

0A　E1　A4

これに**指数**を加えます。指数は，先程と変わっていませんから，8DHです。したがって，

0A　E1　A4　8D

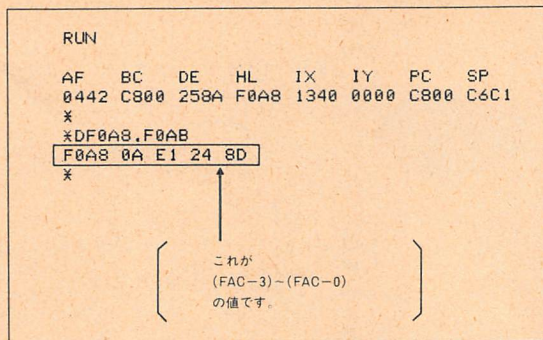
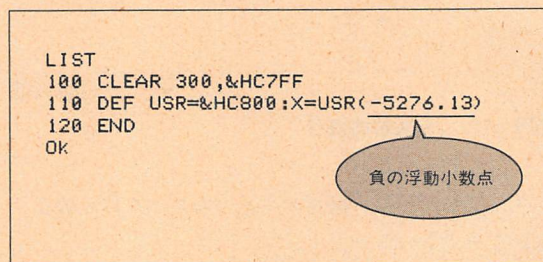
の**4バイト**が浮動小数点アキュームレータに格納されます。

それでは，これもプログラムに入れて確かめてみましょう。第176図の引数部分を

X＝USR（−5276.13）

のように変更します（**第178図**）。マシン語部の準備ができていれば，

RUN↘

でプログラムを走らせましょう。結果は，**第179図**のとおりです。我々の変換どおりの結果ができました。

第177図　浮動小数点アキュームレータを調べる

第178図　負の浮動小数点で実験

浮動小数点型式のまとめ

単精度の浮動小数点については，以上のとおりです。**倍精度**については，**第180図**を御覧ください。図のように，倍精度浮動小数点では，

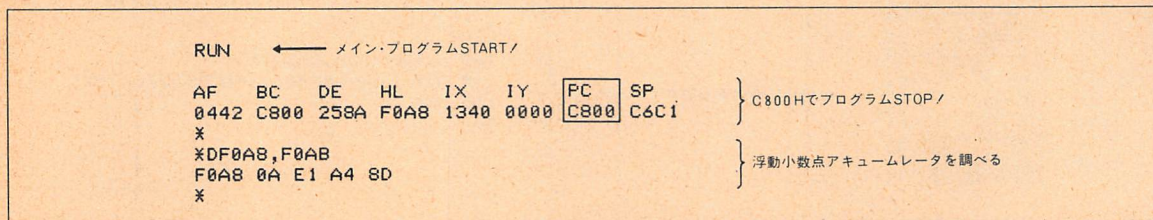
仮数部＝7バイト

に増えています。したがって，2進数に変換するとき4バイト余分に変換します。これにより有効桁数を大幅に増加できます。倍精度の精度が高いのは，このためです。

以上のように仮数部に4バイト多い7バイトの2進数を用意すれば，あとの変換は，**単精度とまったく同様**にして行うことができます。

ここで単精度，倍精度を含めて浮動小数点型式の浮動小数点アキュームレータへの格納方式をまとめておきます。

```
RUN      ←── メイン・プログラムSTART！

AF   BC   DE   HL   IX   IY   PC   SP     } C800Hでプログラム STOP！
0442 C800 258A F0A8 1340 0000 C800 C6C1
*
*DF0A8,F0AB                               } 浮動小数点アキュームレータを調べる
F0A8 0A E1 A4 8D
*
```

第179図　RUN↘

① 浮動小数点（10進数）を**整数部**，**小数部**に分けて2進数に変換する。精度は，**有効数字**が，

単精度＝3×8＝24桁

倍精度＝7×8＝56桁

得られるまで変換を続ける。

② それを有効数字表現

1.mm…m×2 $^{m…m}$

に変換し，小数点部分と指数部分を取り出す。

③ 小数点部分の最上位に**符号ビット**

正数……0
負数……1

を付けた後，**16進数**に変換する。

④ 得られた16進数

単精度……3バイト

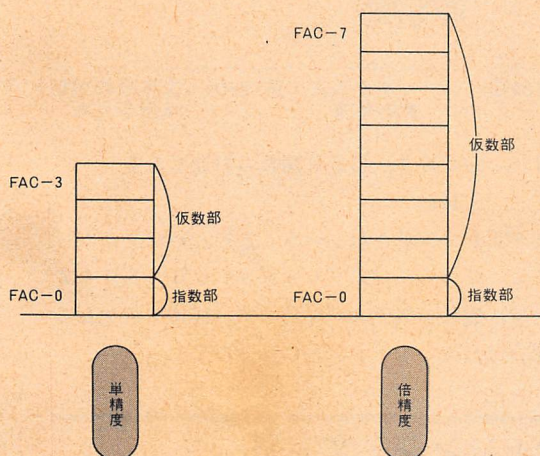
倍精度……7バイト

を逆順にする（**仮数部**）。

⑤ **指数部分**は，単純に1バイトの16進数に変換し，**81H**を加える（**指数部**）。

⑥ ④，⑤で得られた

（仮数部）＋（指数部）

が浮動小数点アキュームレータに格納される値となる。

第180図　浮動小数点アキュームレータ

文字型引数を調べる

浮動小数点型式，いかがでしたか？　脅かしたわりには，簡単でした？　それは，大変結構でした。

ＵＳＲ関数の使い方，さあ，いよいよ最終の

引数＝文字型

の場合について調べていきます。

たとえば，

〝ＡＢＣ〞

という文字例を，メイン・ルーチンからマシン語サブルーチン側に渡したいとします。すなわち

Ｘ＝ＵＳＲ（〝ＡＢＣ〞）

とした時に，マシン語サブルーチン側では**この情報をいかに受け取るか**です。そこで「リファレンス・マニュアル」P.98のこの部分に関する部分を読んでみましょう。

「引き数が文字列の場合には，［ＤＥ］レジスタペアが〝ストリング・デスクリプタ〞と呼ばれる3バイトのデータの番地をポイントします。ストリング・デスクリプタのバイト0が文字列の長さ（0から255まで）を表わし，バイト1とバイト2がそれぞれ文字領域内の文字列の開始番地の下位および上位8ビットを表わします。」

マニュアルにこれだけ丁寧に書かれていれば大体おわかりと思いますが，一応実験を兼ねて説明していきます。

何はともあれ，第178図のプログラムに変更を加えましょう。

Ｘ＝ＵＳＲ（〝**ＡＢＣ**〞）

と，**引数の部分を文字列に変えます**（**第181図**）。

〝ミニ・レジスタ表示プログラム〞等の準備はできていますか？　ＯＫなら

ＲＵＮ↘

```
LIST
100 CLEAR 300,&HC7FF
110 DEF USR=&HC800:X=USR("ABC")
120 END
Ok
```


第181図　文字型引数で実験する

で走らせます。レジスタの値が表示されて，プログラムがSTOPします（第182図）。ここでＤＥレジスタの値に注目してください。

ＤＥ＝ＥＦ５８Ｈ

となっています。これが，

ストリング・デスクリプタ

のアドレスです。

ストリング・デスクリプタ

マニュアルから得られた情報から**ストリング・デスクリプタ**の構造を図式化しますと，**第183図**のようになります。

ストリング・デスクリプタは，図のように**3バイトのエリア**から構成されており，第182図で得られた**ＤＥレジスタ**の値により，

バイト０＝ＥＦ５８Ｈ
バイト１＝ＥＦ５９Ｈ
バイト２＝ＥＦ５ＡＨ

であることがわかります。そこで，この3バイトの値を実際に見てみましょう。

ＤＥＦ５８，ＥＦ５Ａ↘

で表示されます（**第184図**）。

まず**バイト０**を見てみます。

バイト０：ＥＦ５８Ｈ＝**０３Ｈ**

になっています。これが，ＵＳＲ関数引数の

文　字　数

を表わしています。すなわち

〝ＡＢＣ〟＝３文字

というわけです。次の

バイト１：ＥＦ５９Ｈ＝**４０Ｈ**
バイト２：ＥＦ５ＡＨ＝**８０Ｈ**

８０４０Ｈ

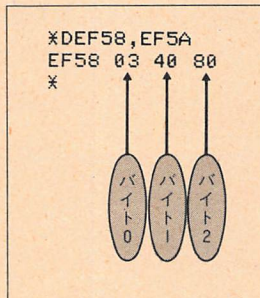
が，実際に**文字列の格納されているアドレス**です。実際，

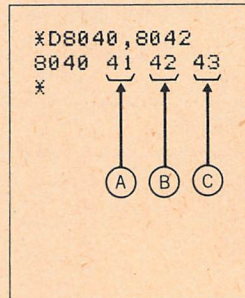
Ｄ８０４０，８０４２↘

でそのアドレスを調べてみますと（**第185図**）

８０４０Ｈ＝４１Ｈ＝〝**A**〟
８０４１Ｈ＝４２Ｈ＝〝**B**〟
８０４２Ｈ＝４３Ｈ＝〝**C**〟

第182図　ファイルデスクリプタのアドレス

第183図　ストリング・デスクリプタ

第184図　ファイルデスクリプタの中身

```
*D8040,8042
8040 41 42 43
*
     Ⓐ  Ⓑ  Ⓒ
```

第184図　文字列の格納状況を見る

のようになっており，ＵＳＲ関数の引数である

〝ＡＢＣ〟

の情報が正しく格納されていることが，わかります。

(注)　ここで得られたアドレス８０４０Ｈ～８０４２Ｈは，**32Ｋシステム**の場合です。**16Ｋシステム**の場合は，異なるアドレスになります。

チャレンジ：文字型

以上が，ＵＳＲ関数の

文字型引数
ファイル・デスクリプタ

の仕組みです。この実験だけでは，文字型引数の使い方が良くわからなかった人のために，次の**チャレンジ**を用意致しました。

> **〈チャレンジ〉**
> メイン・ルーチンから渡された文字列を，マシン語のサブルーチン側で表示するＵＳＲ関数を作りなさい。表示には，
> １文字表示ルーチン：２５７Ｈ
> を使用すること。

それでは，このサブルーチンを考えてみましょう。サブルーチン名は，

PRCHR

と呼ぶことにします。メイン・ルーチンからこのＰＲＣＨＲにプログラムの制御が移ってきた時，

DE＝ファイル・デスクリプタ

の**バイト0**を指しています。そこからの情報を取り出すには，ＨＬレジスタの方が便利ですから，

ＥＸ　ＤＥ，ＨＬ

で値を交換してやります。これで

ＨＬ＝ファイル・デスクリプタ

となります。この最初の１バイト目を，Ｂレジスタに取り出します。

ＬＤ　Ｂ，（ＨＬ）

これで**Bレジスタ**には，

表示すべき文字数

が格納されたことになります。

次に，ＤＥレジスタに実際に文字列が格納されているアドレスを取り出します。

ＩＮＣ　ＨＬ［ＨＬは，アドレスの下位］
ＬＤ　Ｅ，（ＨＬ）
ＩＮＣ　ＨＬ［ＨＬは，アドレスの上位］
ＬＤ　Ｄ，（ＨＬ）

そして，再び

ＥＸ　ＤＥ，ＨＬ

で値を交換します。これで

HL＝文字列の先頭

となったわけです。

すでにＢレジスタには，表示すべき文字数が入っていますから，

ＤＪＮＺでループ

を作り，文字列を表示してやれば良いのです。

〈1文字表示ルーチン〉

２５７Ｈ：Ａレジスタのキャラクタ・コードを表示

でしたから，ループの中身は，

ＬＤ　Ａ，（ＨＬ）［コードを取り出す］
ＣＡＬＬ　２５７Ｈ［表示］
ＩＮＣ　ＨＬ［次の文字へ］

となります。できあがりましたサブルーチンが，**第186図**です。

```
                    ;=================
                    ; PRINT CHRACTERS
                    ;      (82.11.26)
                    ;=================
                    ;
                            ORG   0C800H
                    ;
0257                CRT:    EQU   257H          ;PRINT A
                    ;
C800 EB             PRCHR:  EX    DE,HL         ;HL=STRING DESCRIPTER
C801 46                     LD    B,(HL)        ;CHRACTERS
C802 23                     INC   HL
C803 5E                     LD    E,(HL)
C804 23                     INC   HL
C805 56                     LD    D,(HL)
C806 EB                     EX    DE,HL         ;HL=POINTER OF DATA
C807 7E             PR1:    LD    A,(HL)
C808 CD5702                 CALL  CRT
C80B 23                     INC   HL
C80C 10F9                   DJNZ  PR1
C80E C9                     RET
                    ;
                            END
```

第186図　マシン語サブルーチン

コマンド，ステートメンと関数の違い

メイン・ルーチンは，**第187図**を用います。これは，第181図と同じで，

引数＝〝ABC〞

となっています。それでは，プログラムがうまく動くか走らせてみましょう。

RUN↘

ややっ！　**ピー**と警告音が鳴り，

Type mismatch in 110

と表示されてしまいました（**第188図**）。どこを失敗したのでしょう？

結果を良く見ると，

〝ABC〞の表示には成功

しているのがわかります。その後，メイン・ルーチンに戻ったところでエラーが発生したのですね。エラー表示は，タイプ・ミスマッチが110行で発生したとしています。

Type mismatch

代入文などで式の左右の型が一致していない（数値とストリングなど）。

ですから，110行を見ますと

X＝USR（〝ABC〞）

がいけないということになります。型は，

左辺＝X←――――――――――**単精度**

右辺＝USR（〝ABC〞）←――？

あれっ？　USR（〝ABC〞）の型は何でしょう？

```
100 CLEAR 300,&HC7FF
110 DEF USR=&HC800:X=USR("ABC")
120 END
```

引数

第187図　メイン・ルーチン

第188図　タイプ・ミスマッチ・エラー発生

ところで「N―BASIC　リファレンス・マニュアル」を見ますと，いろいろな命令や関数の説明がされています。そしてよくよく注意してみますと，

第2章　コマンドとステートメント

第3章　関　数

のように**分けて**説明されています。この両者の違いは，何だか御存知ですか？

その決定的な違いは，

コマンド，ステートメント

命令だから，それ自体で**単独使用可**

関　数

命令ではないから，必ず**コマンド，ステートメントと組み合わせて用いる**

ということです。

関数のしくみ

それは，**関数の働き**を考えれば理解できるでしょう。

① 関数は，普通**引数**を持ち，（　）の中に書く［中には，DATE$，TIME$，CSRLINのように引数をもたない関数もあります］。

② 関数が，コマンド，ステートメントと用いて実行されると，すなわち

［LET］ X＝(関　数)

PRINT（関　数）

のように用いられると，関数は**引数の値を使って所定の演算**を開始します。

③ その演算結果は，どこにしまわれるか？

実は，関数自身にしまわれるのです。したがって，

関数は変数のように値を持っており，

そのため型（整数型，…）が存在する

といえます。

④ たとえば，

LET　X＝(関　数)

では，演算結果が関数にしまわれ，その値がXに代入されます。また

PRINT（関　数）

では，関数にしまわれた演算結果が，PRINT文により表示されるというわけです。

⑤ 以上のように関数は，演算結果をしまうため
型……整数型
単精度浮動小数点型式
倍精度浮動小数点型式
文字型
のいずれかを持っています。どの関数がどの型を持つかは，リファレンス・マニュアルで調べてください。

ＵＳＲ関数の型

そこでＵＳＲ関数ですが，型は何でしょう？
答は，通常は

引数の型と同じ

です。たとえば

ＵＳＲ（２１％）←─**整数型**

ＵＳＲ（〃Ａ〃）←─**文字型**

といった具合です。ただし，これはマシン語サブルーチン内で悪いことをしなかった時の話しです。悪いこと──すなわち，マシン語サブルーチン中で変数の型を変えるようなイタズラ（必要なこともありますが）をしない限り

ＵＳＲ関数の型＝引数の型

と考えて結構です。

以上がおわかりになりましたら，第187図の110行を御覧になってください。

ＵＳＲ（〃ＡＢＣ〃）

は文字型ですね。すると，

左辺：単精度≠文字型：**右辺**

ということになり，結局

Type mismatch ／

ということになるわけです。

したがってエラーの発生を押えるのでしたら，左辺を

単精度：Ｘ──→**文字型**：**Ｘ＄**

に変えてやれば良いのです。すなわち，

Ｘ＄＝ＵＳＲ（〃ＡＢＣ〃）

とするのです。**第189図**が変更を加えたプログラムです。

ＲＵＮ↘

しますと，今度は**タイプ・ミスマッチのエラーは発生しません**（**第190図**）。

さて，ついでですから前節の**関数のしくみ**を，もう少しさぐってみることに致しましょう。何をやらかすのかと申しますと，変数Ｘ＄を用意し，

ＵＳＲ関数使用前後で値を比較

してみよう，というわけです。まず使用前としまして

Ｘ＄＝〃デンパ マイコン 9月〃

としておきます。この値を表示させるには，

ＰＲＩＮＴ Ｘ＄

でできます。さらにあとで区分けができるように**コメント**を入れておきましょう最初のＸ＄の値を

Ｘ＄－１（１番目のＸ＄という意味）

ということにして，

ＰＲＩＮＴ 〃Ｘ＄－１：〃；Ｘ＄

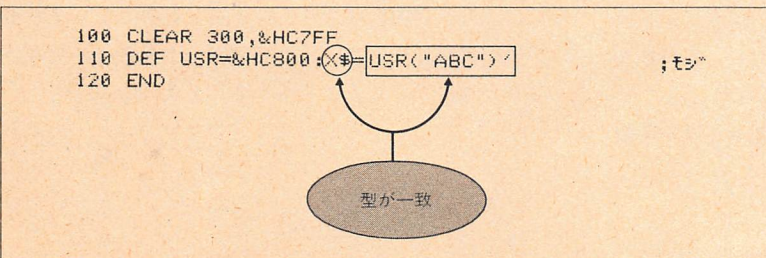
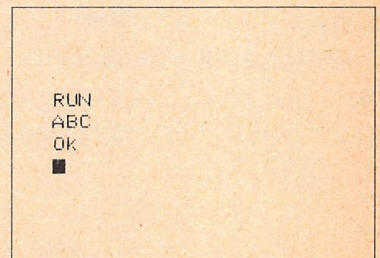
と表示させることにします。

続いてＵＳＲ関数がCALLされ，マシン語サブルーチンの中で引数の値〃ＡＢＣ〃が表示されます。この表示も他と区別するため，

ＰＲＩＮＴ 〃 ＵＳＲ：〃；：

Ｘ＄＝ＵＳＲ（〃ＡＢＣ〃）

とコメントをつけておきます。そして，ＵＳＲ関数からメイン・ルーチンに戻ってくると同時に，その演算結果（←何だと思いますか？）がＵＳＲ関数にしまわれ，その値がさらにＸ＄に代入されます。したがいまして，この段階でＸ＄の値が変化するはずです。この変化した値を調べるため

Ｘ＄－２

第189図 型を一致させて

第190図 今度はエラーなし／

```
100 '==============================
110 '   USRｶﾝｽｳ (ﾓｼﾞｹｲ) ﾉ ｼﾞｯｹﾝ
120 '              ----------82.11.26
130 '==============================
140 '
150 CLEAR 300,&HC7FF:DEF USR=&HC800'            ;SET USRｶﾝｽｳ
160 X$="ﾃﾞﾝﾊﾟ ﾏｲｺﾝｼ 9月"'                      ;X$-1
165 PRINT "X$-1:";X$
166 PRINT " USR:";:X$=USR("ABC"):PRINT'         ;X$-2
167 PRINT "X$-2:";X$
170 END
```

このPRINTは，マシン語サブルーチンの中で改行作業が行なわれないので追加してある。

第191図　ＵＳＲ関数使用前後で？

とコメントをつけ，

PRINT　"X＄－2:"；X＄

で表示させてやります。

USR関数の演算結果

以上の流れをプログラムにまとめたのが，**第 191 図**です。さっそく，走らせてみましょう。

RUN↘

結果は，**第 192 図**に示してあります。御覧のように三行にわたってメッセージが表示されています。

Ⓐ……ＵＳＲ関数実行前のＸ＄の値。

Ⓑ……マシン語サブルーチン内で表示した引数の値。

Ⓒ……ＵＳＲ関数実行後のＸ＄の値。したがってこれは，ＵＳＲ関数にしまわれた演算結果でもある。

以上の実験結果を見ますと，

ＵＳＲ関数にしまわれる値＝引数？

という感じがします。いかがでしょう？

ここで結論を申し上げます。**ＵＳＲ関数にしまわれる演算結果**は，次のようになっています。先にも簡単に触れましたように，ユーザーはマシン語サブルーチン内で**ＵＳＲ関数の型**を決定（変更）することができます（その方法は，本書の続編で紹介するとしましてここでは触れません）。実は**その型によってＵＳＲ関数にしまわれる値が変化する**のです。すなわち，

数　値――現在の浮動小数点アキュームレータの値

文　字――現在のストリング・デスクリプタで示される文字列

という具合になっています。したがって，マシン語サブルーチン内で何もしない場合は，

ＵＳＲ関数にしまわれる値＝引数

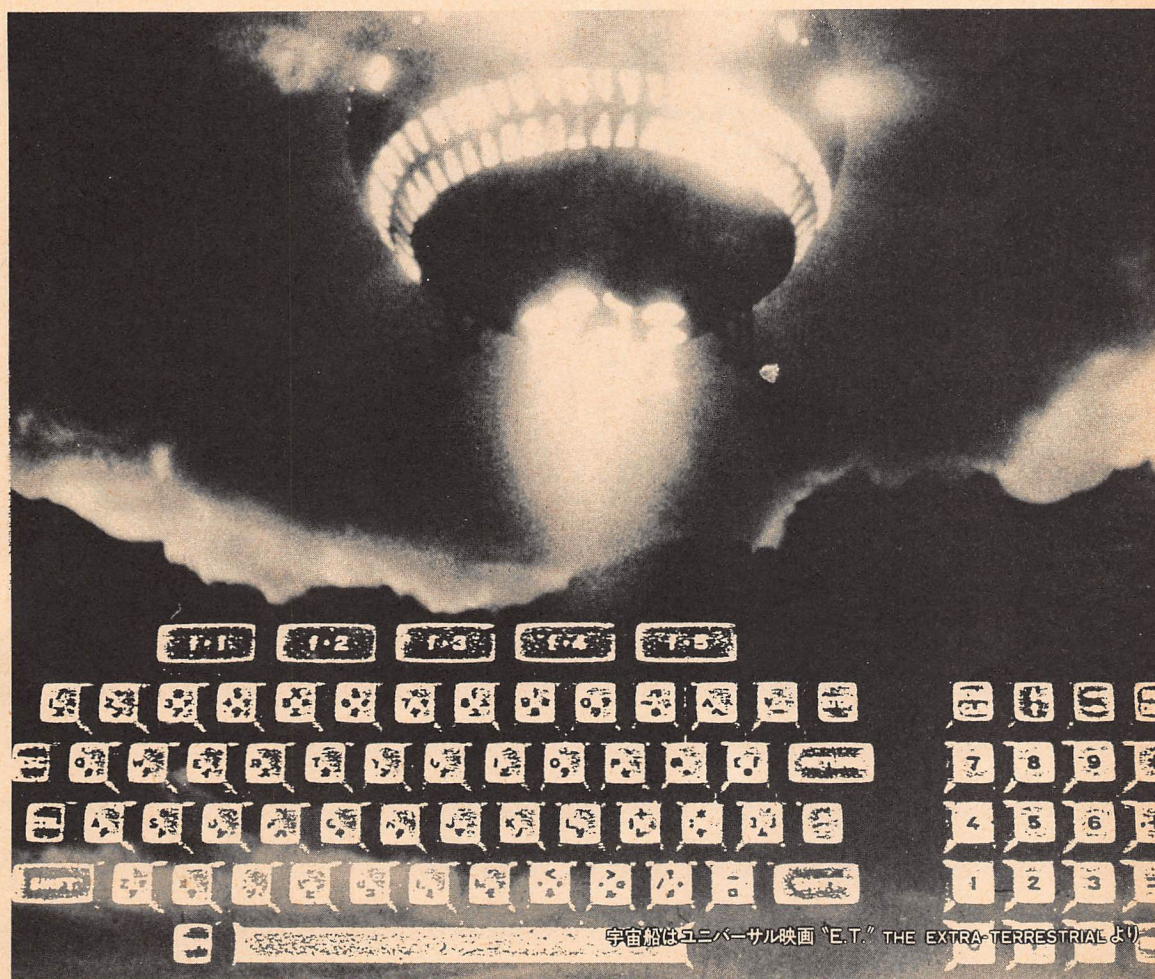
ということになるのです。

以上のように，ＵＳＲ関数には面白い使い方がまだまだいろいろあります。特にマシン語サブルーチン内でＵＳＲ関数の型を変え，浮動小数点アキュームレータやストリング・デスクリプタの値をいじくることで，自由に演算結果をメイン・ルーチンに渡してやることもできます。それを行うには，ちょっと注意（特に文字型の場合）があります。ここでそれを説明していたのでは，ＵＳＲ関数ばかりでページ数が増えてしまいますので続編にゆずり，ここでブロックを変え，次の話題へと進んでいくことに致しましょう。

```
RUN
X$-1:ﾃﾞﾝﾊﾟ ﾏｲｺﾝｼ 9月     ――――― Ⓐ
 USR:ABC                ――――― Ⓑ
X$-2:ABC                ――――― Ⓒ
OK
■
```

第192図　ＲＵＮさせると

第 4 ブロック

効果音付き
カラー・グラフィックの世界

宇宙船はユニバーサル映画"E.T." THE EXTRA-TERRESTRIALより

＜はじめに＞

上手な人の料理は，**盛り付け**も上手で食欲がそそられます。プログラムも同様で，**上手な盛り付け**が必要になります。キャラクタよりは，**グラフィック**を使った方が見栄えが良いし，それが**カラー**になれば，さらに画面効果は盛り上がります。また効果音にしたって，

単純なビープ音

↓

マシン語による電子音

を使った方が，プログラムが引き立ちます。

本ブロックは，主としてこれら

効果を高めるためのテクニック

を中心に話しが展開していきます。すなわち本ブロックの最終目標は，

効果音付き

カラー・グラフィックの実現

です。少し難しいかもしれません。しかし，楽しい部分でもありますから頑張って読み進めていってください。

第13章

カラー・グラフィックの世界

```
E200: 47 20 41 20 4D 20 45 20 20 20 53 20 54 20 41 20 ; 0322
E210: 52 20 54 20 A5 20 A5 20 A5 20 46 20 55 20 4E 20 ; 047E
E220: 43 20 54 20 49 20 4F 20 4E 20 A5 20 35 00 20 47 ; 037E
E230: 20 41 20 4D 20 45 20 20 20 45 20 4E 20 44 20 A5 ; 036F
E240: 20 A5 20 A5 20 A5 20 A5 20 53 20 54 20 4F 20 50 ; 04DA
E250: 20 20 20 4B 20 45 20 59 00 B8 0A 58 50 58 50 00 ; 039B
E260: 1E 04 1F 05 20 06 21 07 22 07 23 07 24 07 25 07 ; 013E
E270: 26 07 27 07 28 06 29 05 2A 04 2B 03 2C 02 2D 01 ; 016F
E280: 00 2C 22 04 2E 22 0C 48 42 08 2C 22 04 2E 22 02 ; 01E4
E290: 94 99 06 1F 11 00 2F 22 0F 87 88 04 9F 99 08 20 ; 0436
E2A0: 2E 00 8E 00 0E 0E 00 0E 48 42 08 2E 42 08 2E 22 ; 0240
E2B0: 02 2E 22 0C 80 8F 00 0F 21 0F 43 48 03 2F 22 0F ; 029A
E2C0: 8F 48 03 9F 99 08 1F 53 08 62 79 20 4B 2E C2 B6 ; 0580
E2D0: BA DE BC 20 69 6E 20 46 4F 52 45 53 49 47 48 54 ; 0616
E2E0: 20 28 31 39 38 31 F2 20 31 31 F3 20 32 39 F4 29 ; 052A
E2F0: 00 98 28 E8 50 00 38 28 E8 50 00 4D 20 59 20 53 ; 04C9
E300: 20 54 20 45 20 52 20 59 00 58 D8 98 00 35 20 30 ; 0411
E310: 20 C3 20 DD 00 32 20 30 20 C3 20 DD 00 31 20 30 ; 04C3
E320: 20 C3 20 DD 00 4B 20 45 20 59 95 95 95 34 00 4B ; 0547
E330: 20 45 20 59 95 95 95 36 00 42 20 45 20 41 20 4D ; 0448
E340: 95 95 95 53 20 50 20 41 20 43 20 45 20 20 20 4B ; 0456
E350: 20 45 20 59 00 D8 0E 88 41 D8 00 B8 13 22 14 A8 ; 050E
E360: 3C 80 3D B8 50 00 31 20 35 20 30 20 30 20 C3 20 ; 042A
E370: DD 20 20 20 A6 20 20 20 BA 20 B4 20 D9 20 C4 CB ; 0679
```

か・ら・ー・だ・ぞ〜〜〜〜

グラフィックの世界に

ここから，しばらくは

グラフィックの世界

をさぐっていくことになります。「PC－8001　マシン語入門（第一巻）」でも，

カラー・グラフィックを実現

するためのヒントを紹介しておきました。中には，もう大分進まれてマシン語による高速グラフィックを楽しんでおられる方もいるようです。また，中には運悪くそこでけっ躓き，足踏みをしておられる方もいるようですね。

ここでは，すべての人が同じ土俵の上に立ち，

グラフィックの実現

に向けて**実験**を進めていくことに致します。

そこで取り上げるカモが，オールマシン語版スペース・インベーダーに出てくる

UFO

です。いずれあとの巻でこれを解析することになりますが，ここで一歩先取りをし，

UFOの動き

を追ってみることにします。

グラフィック・データへの変換

あのインベーダーの中に出てくる**UFOのデザイン**は，**第193図**のようになっています。グラフィックは，

1キャラクタ＝2×4ドット

のように分割して使われていますので（**第194図**），

UFOの大きさ＝9×2

ということになります。

まずこの図形を

グラフィック・コード

に変換します。変換の仕方をまとめますと（**第195図参照**），

① 1キャラクタを**左右に分割**します。

② 各分割は，**4ドット**から構成されています。そして各ドットについて

白い部分……0

黒い部分……1

のように変換します。

③　その4桁の0と1を図のように並べ換えます，すると**4ビットの2進数**ができあがります。

④　それを**16進数**に変換します。

⑤　左右に得られた16進数を**逆順**にします。

⑥　⑤で得られた1バイトの数が，

グラフィック・コード

となります。

この変換のルールにしたがってUFOの図形をグラフィック・コードに変換していきますと，

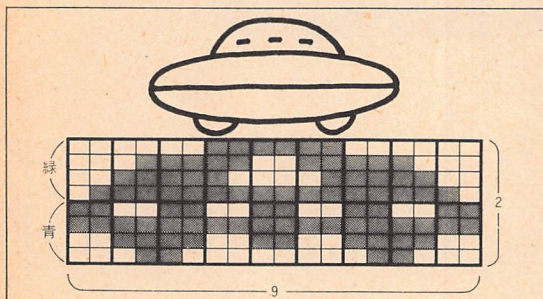
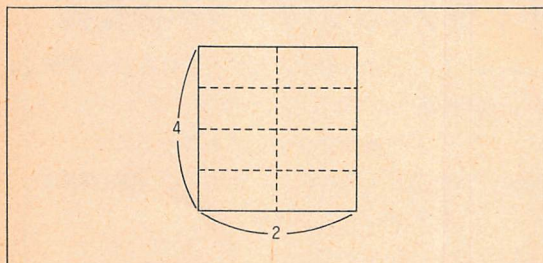
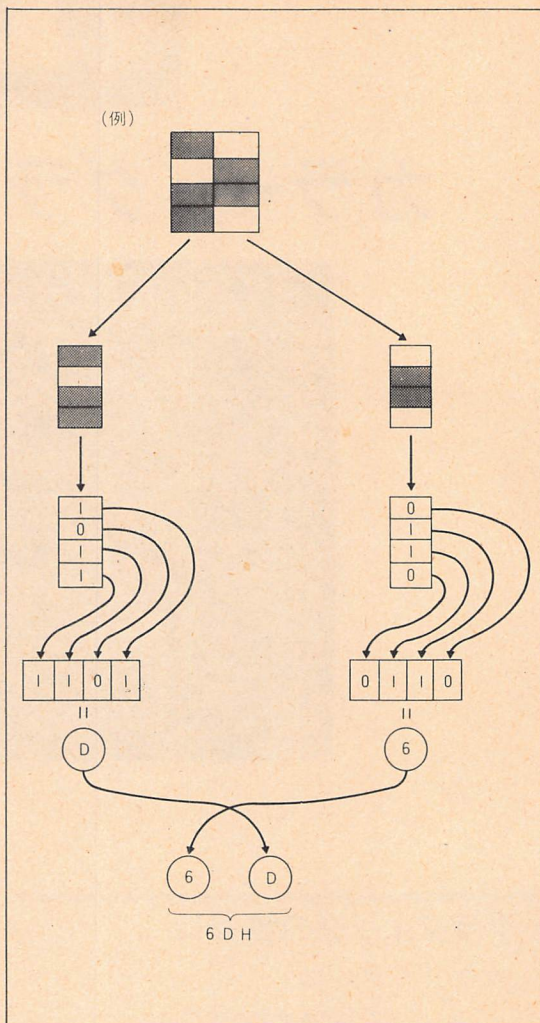
1キャラクタの大きさ＝1バイト

の割でコードに変換されますから，**第196図**のように

9×2＝18（バイト）

のグラフィック・コードが得られます。これをアセンブラの**DB命令**で定義したのが，**第197図**です。データの先頭には，

DUFO

と名付けてあります。

第193図　UFOのグラフィック

第194図　1キャラクタ＝4×2ドット

第195図　グラフィック・コードに変換する

80	EC	EE	BF	99	FB	EE	CE	08
33	E6	7F	22	77	22	F7	6E	33

第196図　UFOのグラフィック・データを得る

```
C81E 80ECEEBF DUFO:  DB   80H,0ECH,0EEH,0BFH,99H,0FBH,0EEH,0CEH,8
C822 99FBEECE
C826 08
C827 33E67F22        DB   33H,0E6H,7FH,22H,77H,22H,0F7H,6EH,33H
C82B 7722F76E
C82F 33
```

UFOの
グラフィック・データ

第197図　UFOのデータ部

UFO表示ルーチンの作成

次にこれを

BASIC＋マシン語

で表示することを考えてみます。すなわち，

BASIC部：UFOの位置を指定

マシン語部：指定された位置にUFOを表示

という具合です。そこで，まずそのマシン語部から見ていくことに致しましょう。おっと，

「キャラクタ・コードをTV画面に表示する方法なら知っているが，グラフィック・コードをTV画面に表示する仕方はまだ知らないぞ！」

とおこられるかもしれません。しかし，有難いことに

グラフィック・コードのTV画面への表示法は，キャラクタ・コードと同様

に行うことができます。すなわち

ビデオRAM←グラフィック・コード

のように転送を行えばよいのです。

そうしますと，UFOの表示は，第171図で行った**ビーム砲の表示と同様**に行うことができることになります。すなわち，

```
LD   E, (HL)   ┐Y座標
INC  E         ┘
INC  HL
LD   D, (HL)   ┐X座標
INC  D         ┘
EX   DE, HL
```

でHLレジスタにLOCATE座標を取り出し，

```
CALL LOC
```

でビデオRAMのアドレスに変換します。そして，

```
LD      DE, DUFO
```

```
                   ;=================
                   ;  PRINT UFO
                   ;      (82.11.25)
                   ;=================
                   ;
                           ORG   0C800H
                   ;
03F3               LOC:    EQU   3F3H             ;LOCATE TO ADDRESS
                   ;
C800 5E            PUFO:   LD    E,(HL)           ;PRINT UFO FROM BASIC
C801 1C                    INC   E                ;(1,1) ORIGIN
C802 23                    INC   HL
C803 56                    LD    D,(HL)           ;DE=LOCATE OF UFO
C804 14                    INC   D
C805 EB                    EX    DE,HL
C806 CDF303                CALL  LOC
C809 111EC8                LD    DE,DUFO          ;UFO DATA
C80C CD0FC8                CALL  PU1
                   ;
C80F E5            PU1:    PUSH  HL               ;IN(DE=DATA,HL=ADDRESS)
C810 0609                  LD    B,9              ;9 BYTES PER 1 LINE
C812 1A            PU2:    LD    A,(DE)
C813 77                    LD    (HL),A
C814 13                    INC   DE
C815 23                    INC   HL
C816 10FA                  DJNZ  PU2
C818 E1                    POP   HL
C819 017800                LD    BC,120           ;HL=HL+120
C81C 09                    ADD   HL,BC
C81D C9                    RET
                   ;
C81E 80ECEEBF      DUFO:   DB    80H,0ECH,0EEH,0BFH,99H,0FBH,0EEH,0CEH,8
C822 99FBEECE
C826 08
C827 33E67F22              DB    33H,0E6H,7FH,22H,77H,22H,0F7H,6EH,33H
C82B 7722F76E
C82F 33
                   ;
                           END
```

第198図　UFO表示ルーチン

でUFOのデータの先頭アドレスをDEレジスタにセットし，**PLINEを2回CALL**（2行分だから）してやれば良いのです。こうしてできあがったマシン語サブルーチンが，**第198図**です。

(注) 1．第198図では，PLINEがPU1になっています。また1行の長さがUFOに合わせて変更されています。

2．第171図では一行表示毎にデータ・アドレスの位置をセットし直しましたが，実際は必要ありませんので，第198図では省略してあります。

キャラクタが出現！

次に**BASIC部のメイン・ルーチン**です。

CLEAR　300，&HC7FF

でBASICの**使用領域に制限**を加え，

DEF　USR＝&HC800

で**USR関数を定義**します。また

DEFINT　A－Z

で**変数を整数化**しておきます。これは，浮動小数点アキュームレータを**整数型**として使うためです。UFOの座標を表わす変数として

ヨコ座標——X

タテ座標——Y

を使うとして，とりあえず

LOCATE　0，2

の位置に表示してみましょう。すると，

X＝0：Y＝2

と設定してから，USR関数をCALLすることになります。その前に

XY＝X＊256＋Y

で**2バイトの数**に変換するのでしたね？　今度は，ほんのちょぴっと高級(?)に

X＊256＋Y

↓ 直接引数に！

U＝USR（　　　　　）

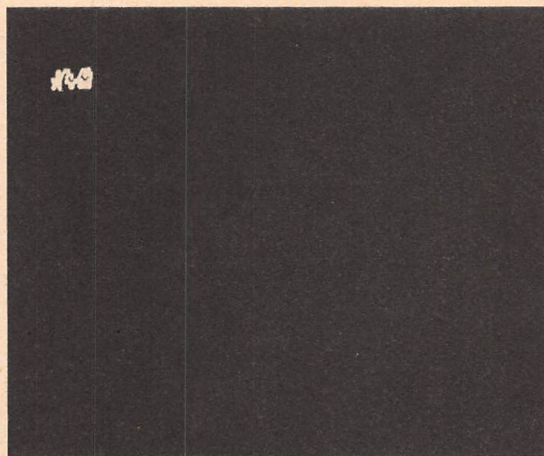
のように変換式を直接USR関数の引数に使ってみましょう。すなわち，

U＝USR（X＊256＋Y）

のようにしてUSR関数を呼び出します。こうしてできあがったメイン・ルーチンが**第199図**です。

《写真14》 変なキャラクタ出現！

```
1000 '================================
1010 '  MOVING UFO-1
1020 '     1982.12.1:BY K.TSUKAGOSHI
1030 '================================
1040 '
1050 CLEAR 300,&HC7FF:DEF USR=&HC800:DEFINT A-Z' ;SET USR FUNCTION
1060 WIDTH 80,25:CONSOLE 0,25,0,0'              ;SET TV MODE
1070 COLOR 0:PRINT CHR$(12)
1080 '
1090 X=0:Y=2:U=USR(X*256+Y)
1100 GOTO 1100                                 ;LOOP FOREVER
```


第199図　UFO表示のBASICメイン・ルーチン

それでは，さっそくプログラムを走らせましょう。第198図，第199図の両方のプログラムを入力したら，

RUN↘

します（**写真14**）。

〝ヤヤッ！　**変なキャラクタが**〟

ＴＶ画面，左上に現われました。ちょうど**ＵＦＯの位置**です。

グラフィック・モード

この原因は，ハッキリしています。我々がやったのは，

（転送）
ビデオRAM←——UFOの
グラフィック・コード

でした。しかし，PC-8001のハードウェアは，

ビデオRAM←—キャラクタ・コード

と解釈し，**該当するキャラクタ・コードを表示してしまった**のです。ちなみに我々の用意したグラフィック・コードは，

80H，ECH，EEH，……

でした。これをキャラクタ・コード表に照らし合わせてみますと，

80H＝▁
ECH＝●
EEH＝▨
⁝

のようになります。TV画面には，これらのキャラクタ・コードが表示されてしまったのです。

それでは，どうしたらTV画面に

グラフィックを表示する

ことができるでしょうか？

ここで「ユーザーズ・マニュアル」のP.26を御覧ください。「2.6　COLOR」の説明の下の方です。

〝〈グラフィックスイッチ〉を1にすればグラフィックモードに，0にすればキャラクタモードになります〟と説明されています。

すでに御存知のこととは思いますが，PC-8001の画面モードには，

〈**キャラクタ・モード**
　グラフィック・モード

の二種類があります。この両者を切り換えるのが

COLOR ①，②，③
↑
第3パラメータ

です。

もう，おわかりでしょう。画面モードを**COLOR文**で

グラフィック・モード

に切り換えれば，

ビデオRAM←—グラフィック・コード

の転送で，正しくグラフィックが表示されるのです。

グラフィックUFOに成功

画面モードをグラフィック・モードに切り換えるには，

COLOR　0，0，1

のようにCOLOR文の**第3パラメータを1**にすればよいのです（第1，第2パラメータについては，直接ここでの話題とは関係ありませんし，BASICの話しなので，説明致しません。御存知ない方は，マニュアルで調べてください）。

（注）　実はCOLOR文の第3パラメータを1にしただけでは，グラフィック・モードにはなりません。この命令だけでは，ワーク・エリア中の画面モードを表わす値が，グラフィックを示すようになるだけです。その後，いつでも結構ですが

PRINT　CHR$（12）

のように**画面を消去**すると，アトリビュート・エリアにグラフィックを表わすアトリビュート・コードが転送され，グラフィック・モードになります。

そこでTV画面をグラフィック・モードにするために，第199図のプログラムに手を加えます。

```
1070 COLOR 0，0，1：
     PRINT CHR$（12）
```

とすれば良いですね？　できあがったプログラムが，**第200図**です。マシン語サブルーチンには，手を加える必要はありません。

さっそく

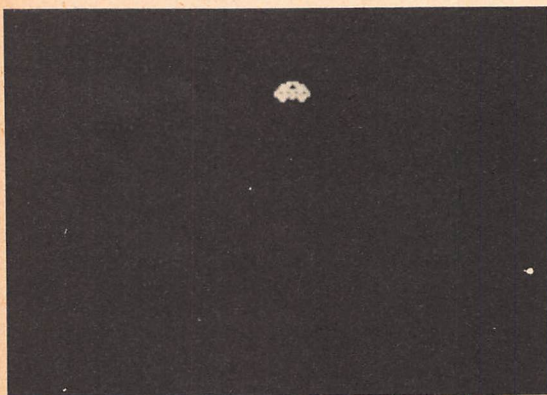
RUN↘

してみましょう。**写真15**のとおりです。やりました！ついに

```
1000 '================================
1010 '  MOVING UFO-2
1020 '     1982.12.1:BY K.TSUKAGOSHI
1030 '================================
1040 '
1050 CLEAR 300,&HC7FF:DEF USR=&HC800:DEFINT A-Z'  ;SET USR FUNCTION
1060 WIDTH 80,25:CONSOLE 0,25,0,0'                ;SET TV MODE
1070 COLOR 0,0,①:PRINT CHR$(12)'                  ;SET COLOR GRAPHIC
1080 '
1090 X=0:Y=2:U=USR(X*256+Y)
1100 GOTO 1100'                                   ;LOOP FOREVER
```

これで
グラフィック・モード
にすることができる

第200図　グラフィック・モードに変更するために

グラフィックでＵＦＯを表示！
させることに成功したのです。

《写真15》 グラフィックでＵＦＯ完成

カラー・グラフィックに挑戦

これであなたは，

ＢＡＳＩＣ＋マシン語による

高速グラフィックGAME

が自由に作れるようになったのです。これは事実で，これだけの知識でも，

アイデアと努力！

により，かなり高級なプログラムを作り上げることができるのです。そして，実際にそのような作品がすでに存在しています。さあ，あなたもひと奮発してみてはいかがでしょうか？

さて，せっかくここまできたのですから，

カラー・グラフィック

もあやつってみたいとは思いませんか？　そこで次に挑戦するのが，

カラー・グラフィックによるＵＦＯ

の実現です。

先に

キャラクタ・モード↔グラフィック・モード

の切り換えは，ＣＯＬＯＲ命令の第3パラメータにより，**グラフィック・スイッチ**（「ＰＣ－8001　ユーザーズ・マニュアル」P.20）を切り換えることで行いました。ここで**画面モード**を別の観点からとらえてみますと，

{ 白黒モード
{ カラーモード

の二つのモードが存在することがわかります。

もうお気付きのことと思いますが，

白黒モード↔カラー・モード

の両者を切り換えることで，**カラー**を楽しむことができるのです。この両者を切り換えるのが，

ＣＯＮＳＯＬＥ命令の第4パラメータ

で，

ＣＯＮＳＯＬＥ　①，②，③，１

とすることでカラー・モードにすることができます。

まずはCOLOR文で

以上のようにＣＯＮＳＯＬＥ命令を使って第200図のメイン・ルーチンを書き換えたのが，**第201図**です。1060行を御覧ください。

ＣＯＮＳＯＬＥ　０，２５，０，１
↑
（カラー・モード）

となっています。

```
1000 '================================
1010 '  MOVING UFO-3
1020 '     1982.12.1:BY K.TSUKAGOSHI
1030 '================================
1040 '
1050 CLEAR 300,&HC7FF:DEF USR=&HC800:DEFINT A-Z'  ;SET USR FUNCTION
1060 WIDTH 80,25:CONSOLE 0,25,0,①'               ;SET TV MODE
1070 COLOR 4,0,1:PRINT CHR$(12)'                 ;SET COLOR GRAPHIC
1080 '    (グリーン)
1090 X=0:Y=2:U=USR(X*256+Y)
1100 GOTO 1100'                                  ;LOOP FOREVER
```

カラーモード

第201図　カラー・モード（緑）に

ところで，カラー・モードにするのは良いのですが，肝心の

色の付け方

は，どうしたら良いのでしょう？

この最も簡単な方法は，**ＣＯＬＯＲ命令の第１パラメータ**を使う方法です。たとえば，グリーンで表示したければ，

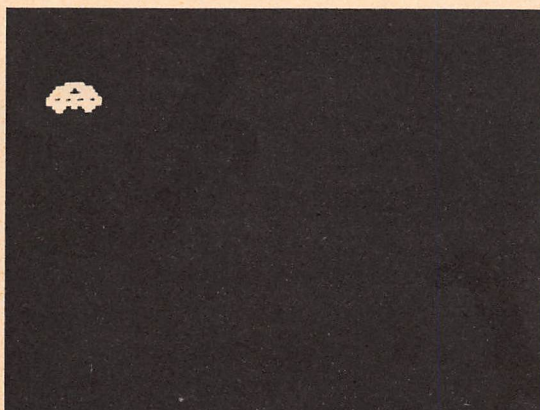
ＣＯＬＯＲ　４，０，１
　　　　　　↑　　↑
（グリーン）（グラフィック指定）

のように設定し，

ＰＲＩＮＴ　ＣＨＲ＄（１２）

を行えばできます。

第201図のプログラムでも，1070行でそれを行っていますね？　それでは，プログラムを走らせてみましょう。第201図のメイン・ルーチン，また第198図のマシン語サブルーチンは入っていますか？

ＲＵＮ↘

《写真16》　緑色のＵＦＯ完成

結果は，**写真16**のとおりです。写真は白黒になっていますが，あなたのカラーＣＲＴには

緑色のＵＦＯ

が鮮かに表示されているものと思います。

LINE文，書式3

ここで，もう一度第193図を御覧ください。ＵＦＯのグラフィック・デザインです。オール・マシン語版スペース・インベーダーは，この図のように

上　部：**グリーン**
下　部：**ブルー**

の**二色**で表示しています。ところが，前節の実験では，

オール・グリーンのＵＦＯ

が表示されてしまいました。これは，色の指定をＣＯＬＯＲ文で行ったからです。ＣＯＬＯＲ文を使うと，どうしても

画面全体が同一色に統一

されてしまうのです。

そこで別の方法を考えてみることに致します。登場するのが，**ＬＩＮＥ文**。第一巻では，方法だけを暗示しておきましたが，その具体的な使い方を見ていきます。

ところでＬＩＮＥ文には，

① カラー・モードで一行単位に機能を指定。
② キャラクタ表示での利用。
③ グラフィックでの利用。

の三種類があります。もちろん，ここでは③を利用することになります。「リファレンス・マニュアル」では，P.44の書式３に当たります。すなわち

LINE　(X, Y) − (x, y)
　, PRESET, 色番号, BF

としてやれば,

指定範囲を, 指定色で, ドットでクリア

することになります。これを**アトリビュート・エリアの立場**から見ると,

指定したエリアを指定した色で
グラフィックに設定

したことになるのです。LINE文のこの機能を用いれば, 画面の仕意の位置を任意の色に設定できます。

オール・マシン語版と同じ

そこでLINE文を使うために,

UFOの移動位置

をグラフィック (ドット) 単位ではかります。

第202図を御覧ください。Ⓐが, **キャラクタ単位**で示したUFOの移動位置です。

タテ座標＝2〜3

の位置を左右に移動させるとします。それをグラフィック単位で示したのが, 同図のⒷです。UFOは,

タテ座標＝8〜15

の位置を左右に動くことになります。ところで,

UFO ｛上　部……グリーン
　　　 下　部……ブルー

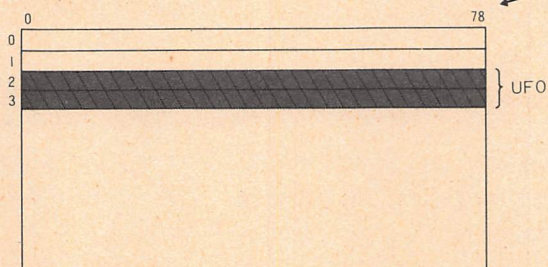
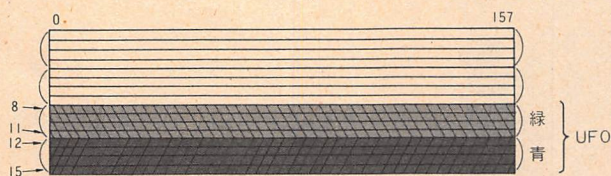
でしたから,

タテ座標＝8〜11：グリーンに設定
　　　　＝12〜15：ブルーに設定

しなければなりません。ヨコ座標は, 左右目いっぱいに動かすとして, LINE文を

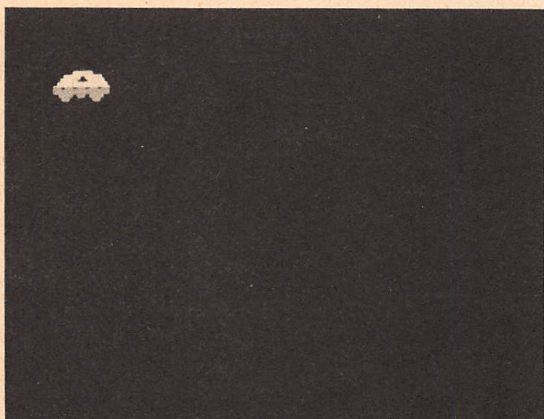
LINE　(0, 8) − (157, 11)
　, PRESET, 4, BF………**緑**に設定
LINE　(0, 12) − (157, 15)
　, PRESET, 1, BF………**青**に設定

のように用いれば良いことがわかります。

以上の計算のもとに, 第201図のメイン・ルーチン

第202図　UFOの移動範囲

```
1000 '================================
1010 '  MOVING UFO-4
1020 '     1982.12.1:BY K.TSUKAGOSHI
1030 '================================
1040 '
1050 CLEAR 300,&HC7FF:DEF USR=&HC800:DEFINT A-Z' ;SET USR FUNCTION
1060 WIDTH 80,25:CONSOLE 0,25,0,1'               ;SET TV MODE
1070 COLOR 7,0,0:PRINT CHR$(12)'                 ;SET COLOR GRAPHIC
1080 '
1090 LINE (0, 8)-(157,11),PRESET,4,BF'           ;GREEN
1100 LINE (0,12)-(157,15),PRESET,1,BF'           ;BLUE
1110 X=0:Y=2:U=USR(X*256+Y)
1120 GOTO 1120'                                  ;LOOP FOREVER
```

カラーの設定

第203図　メイン・ルーチンをLINE文で書き換える

《写真17》　２色のＵＦＯ完成

を書き換えたのが，**第203図**です。さっそく，プログラムを入力し，走らせてみましょう。

RUN↘

結果は，**写真17**のとおりです。またまた白黒でハッキリお見せできませんが，

UFOが二色

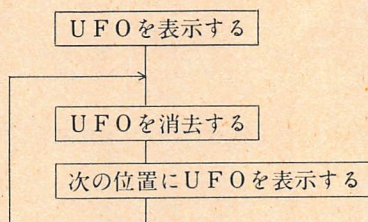
で表示されました。ついにあなたの手で，オール・マシン語版スペース・インベーダーとまったく同じＵＦＯを，あなたのＣＲＴ上に表示させることに成功したのです！

何も表示されないグラフィック・コード

さあ，せっかく

カラー・グラフィック版ＵＦＯ

に成功したのです。これから，これを動かしてみることに致しましょう。

現在，配色は第202図Ⓑのように設定されています。したがいまして，この範囲内で左右に動かす分には，**色の**心配は不要です。ですから，ＵＦＯを動かすことだけに専念すれば良いのです。図形の動かし方なら，もうあなたは得意中の得意ですね？　アルゴリズムは，

を採用することにします。

現在，マシン語サブルーチン中には，

ＵＦＯ表示ルーチン

しか入っていません。したがって

UFO消去ルーチン

を追加しなければなりません。サブルーチン名は，

EUFO

とつけることにします。

ＥＵＦＯは，ＰＵＦＯと同じくＢＡＳＩＣのサブルーチンから呼び出されます。ですから，まず

```
LD      E, (HL)
INC     E
INC     HL
LD      D, (HL)
```

INC　D
EX　DE, HL
CALL　LOC

でHLレジスタに，UFOの左肩の位置（ビデオRAMのアドレス）を得ます。ここから，右下に

9×2

の大きさを消去すれば良いのです。それには，

何も表示されないグラフィック・コード

をビデオRAMに転送してやれば良いのです。

UFO消去ルーチン

```
                   ;=================
                   ;  PRINT UFO-2
                   ;      (82.12.3)
                   ;=================
                   ;
                          ORG  0C800H
                   ;
03F3               LOC:   EQU  3F3H            ;LOCATE TO ADDRESS
                   ;
C800 5E            PUFO:  LD   E,(HL)          ;PRINT UFO FROM BASIC
C801 1C                   INC  E               ;(1,1) ORIGIN
C802 23                   INC  HL
C803 56                   LD   D,(HL)          ;DE=LOCATE OF UFO
C804 14                   INC  D
C805 EB                   EX   DE,HL
C806 CDF303               CALL LOC
C809 113AC8               LD   DE,DUFO         ;UFO DATA
C80C CD0FC8               CALL PU1
                   ;
C80F E5            PU1:   PUSH HL              ;IN(DE=DATA,HL=ADDRESS)
C810 0609                 LD   B,9             ;9 BYTES PER 1 LINE
C812 1A            PU2:   LD   A,(DE)
C813 77                   LD   (HL),A
C814 13                   INC  DE
C815 23                   INC  HL
C816 10FA                 DJNZ PU2
C818 E1                   POP  HL
C819 017800               LD   BC,120          ;HL=HL+120
C81C 09                   ADD  HL,BC
C81D C9                   RET
                   ;
C81E 5E            EUFO:  LD   E,(HL)          ;ERASE UFO
C81F 1C                   INC  E
C820 23                   INC  HL
C821 56                   LD   D,(HL)
C822 14                   INC  D
C823 EB                   EX   DE,HL           ;HL=LOCATE OF ERASE UFO
C824 CDF303               CALL LOC
C827 AF                   XOR  A               ;FOR ERASE CODE  (Aレジスタ=00H)
C828 0E02                 LD   C,2             ;2 LINES
C82A 117800               LD   DE,120          ;FOR HL=HL+120
C82D E5            EU1:   PUSH HL
C82E 0609                 LD   B,9
C830 77            EU2:   LD   (HL),A
C831 23                   INC  HL
C832 10FC                 DJNZ EU2
C834 E1                   POP  HL
C835 19                   ADD  HL,DE
C836 0D                   DEC  C
C837 20F4                 JR   NZ,EU1
C839 C9                   RET
                   ;
C83A 80ECEEBF DUFO:  DB   80H,0ECH,0EEH,0BFH,99H,0FBH,0EEH,0CEH,8
C83E 99FBEECE
C842 08
C843 33E67F22        DB   33H,0E6H,7FH,22H,77H,22H,0F7H,6EH,33H
C847 7722F76E
C84B 33
                   ;
                          END
```

UFO消去ルーチン

第204図　ＵＦＯ消去ルーチンを追加する

そのコードとは，００Ｈです。なぜなら，もう一度第195図を御覧になってください。ここで**すべてのドットを白**にすれば，消去することができます。すると，

すべてのビットが0

になりますから，**００Ｈ**が得られます。

そこで，その**００ＨをＡレジスタにセット**しましょう。論理演算を用いて

```
XOR A
```

でＯＫです。このグラフィック・コードを

```
LD  (HL), A
```

でビデオＲＡＭに転送してやれば，消去できます。ＵＦＯを一行分消去するなら，

```
    LD    B, 9   (9キャラクタ分)
Ⓐ:LD    (HL), A
    INC   HL
    DJNZ  Ⓐ
```

でＯＫです。その前後を

```
┌PUSH HL
│  [一行消去のループ]
└POP  HL
```

のように**PUSH**，**POP**ではさみ，

```
LD    DE, 120
ADD   HL, DE
```

とすれば，**ＨＬは次の行の先頭**に移ります。

以上の処理を，

```
    LD   C, 2
Ⓨ: [一行消去
    HLを次の行の先頭に]
    DEC  C
    JR   NZ, Ⓨ
```

のように二回繰り返してやれば，ＵＦＯ全体を消去してやることができます。

以上の，ＵＦＯ消去ルーチンを組み込みましたマシン語サブルーチンが，**第204図**です。

UFOを右へ

次にメイン・ルーチンです。とりあえずＵＦＯを左右に動かしてみましょう。

今度は，先にできあがったプログラムを御覧いただきましょう。**第205図**です。まず，

USR0：ＵＦＯの表示
USR1：ＵＦＯの消去

のようにＵＳＲ関数を割り当てます（1060行）。そして，ＵＦＯを**最初の位置**

（０，２）

に表示します（1130行）。1140行～1160行が，**ＵＦＯを右に動かすルーチン**です。

（X，Y）のUFOを消去

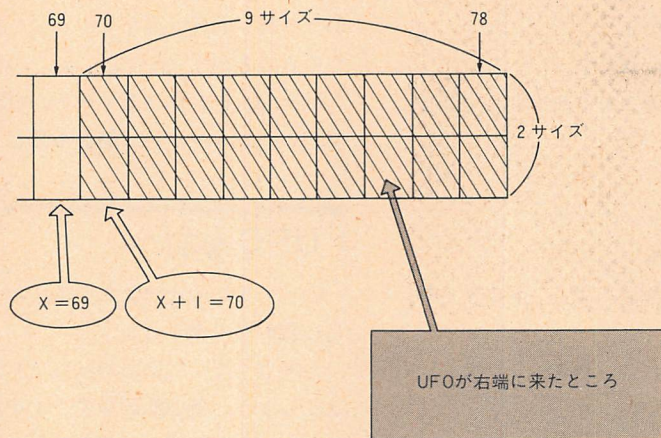
Ｕ＝ＵＳＲ（Ｘ＊２５６＋Ｙ）

（X＋1，Y）にUFOを表示

Ｕ＝ＵＳＲ（（Ｘ＋１）＊２５６＋Ｙ）

をＸの値が **0（左端）**から**69（右端）**まで繰り返します。1160行の

```
1000 '================================
1010 '  MOVING UFO-5
1020 '      1982.12.4:BY K.TSUKAGOSHI
1030 '================================
1040 '
1050 CLEAR 300,&HC7FF:DEFINT A-Z'                  ;SET USR FUNCTION
1060 DEF USR0=&HC800:DEF USR1=&HC81E
1070 WIDTH 80,25:CONSOLE 0,25,0,1'                 ;SET TV MODE
1080 COLOR 7,0,1:PRINT CHR$(12)'                   ;SET COLOR GRAPHIC
1090 '
1100 LINE (0, 8)-(157,11),PRESET,4,BF'             ;GREEN
1110 LINE (0,12)-(157,15),PRESET,1,BF'             ;BLUE
1120 '
1130 X=0:Y=2:U=USR0(X*256+Y)'                      ;SET UFO AT LOCATE 1,2
1140 FOR X=0 TO 69'                                ;MOVE UFO RIGHT
1150   U=USR1(X*256+Y):U=USR0((X+1)*256+Y)
1160   FOR J=0 TO 50:NEXT'                         ;TIMER
1170 NEXT
1180 FOR X=70 TO 1 STEP -1'                        ;MOVE UFO LEFT
1190   U=USR1(X*256+Y):U=USR0((X-1)*256+Y)
1200   FOR J=0 TO 50:NEXT'                         ;TIMER
1210 NEXT:GOTO 1130'                               ;LOOP FOREVER
```

第205図　ＵＦＯを左右に（メイン・ルーチン）

第206図　ＵＦＯが右端に来ると

ＦＯＲ　Ｊ＝０　ＴＯ　５０：ＮＥＸＴ
は，ＵＦＯの動きを鈍くするためのタイマーです。さもないと，表示や消去ルーチンがマシン語ゆえ，ＵＦＯの動きが速すぎてしまうのです。ところで

ＵＦＯの右端：Ｘ＝６９

というのは，おかしいと思いませんか？

これも良く図を書いて考えれば，正しいことがわかります（**第206図**）。画面は，ヨコ＝80サイズですが，ＬＯＣＡＴＥ座標は，

左端を０と数える＝**０オリジン**

ですから，

右端：Ｘ＝７９

です。ところが，カラー・モードではヨコ・サイズが一つ減りますから

右端：Ｘ＝７８

となります。さて，ＵＦＯのサイズは

９×２

ありますから，ＵＦＯが右端まで来ると，

ＵＦＯの左肩：Ｘ＝７０

です。あれ？　Ｘ＝６９　となっていませんね？

でもこれで良いのです。メイン・ルーチンの1140行を見てください。ＦＯＲ～ＮＥＸＴループで，最後はＸの値が69となっていますが，ＵＳＲ関数の引数として

Ｘ＝６９：ＵＦＯを消去

Ｘ＋１＝７０：ＵＦＯを表示

していますね？

第13章のおわりに

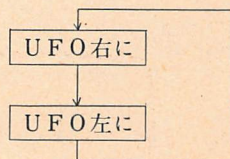
逆に，**ＵＦＯを左に動かす**ときは，

Ｘ：１←―７０

のようにループを組みます（1180行～1210行）。

そして，最後に

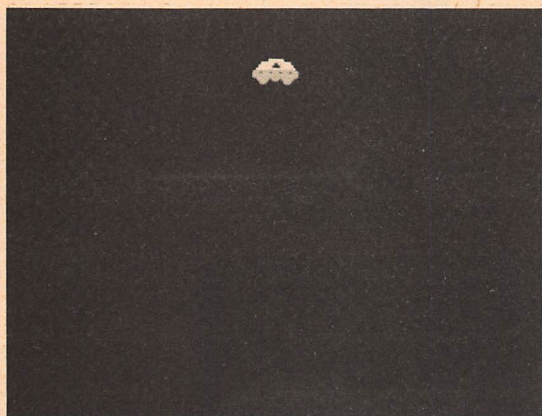
ＧＯＴＯ　１１３０

としてやれば，

の無限ループができ上がります。

それでは，でき上がったプログラムを走らせましょう。

｛ＢＡＳＩＣメイン・ルーチン：第205図
　マシン語サブルーチン：第204図

の両者をキー・インしてください。そして，

ＲＵＮ↘

《写真18》 ＵＦＯが左右に移動する

です。**写真18**のようにＵＦＯは，

右へ行ったり，左に戻ったり

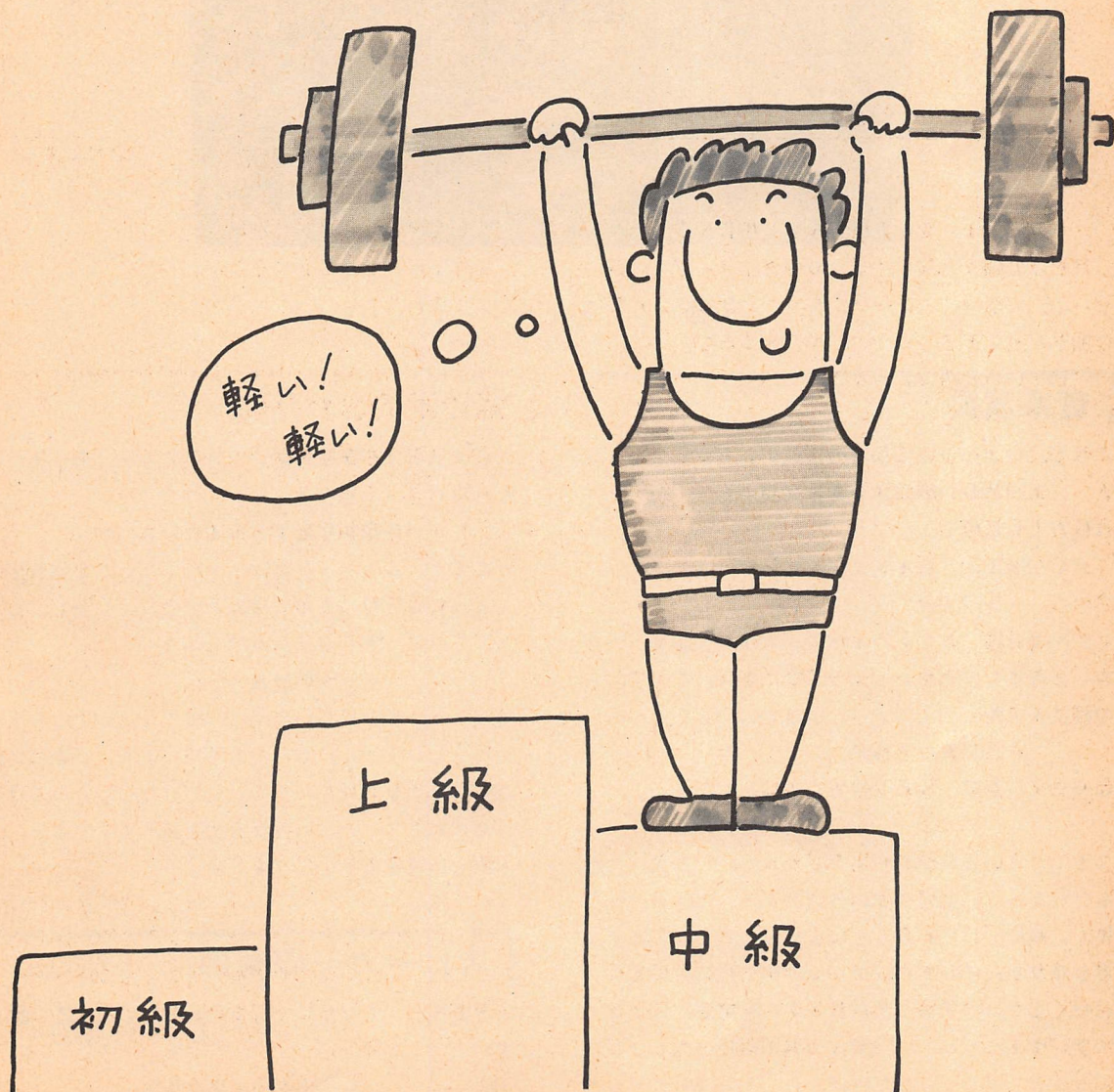
を繰り返します。

いかがでしたか？ **カラー・グラフィックの世界**，意外と簡単でしょう？ あなたが初めて「ＰＣ－8001マシン語入門」（第一巻）を手にした時，ここまで来られるか不安であったかもしれません。でも，あなたはとうとうここまでたどりついたのです。さらば，もうひと踏ん張り。第二巻の最後に

マシン語による電子音

に挑戦することに致しましょう。

軽い！
軽い！

上級
中級
初級

第14章

音楽演奏に挑戦

```
E380: DE 20 B0 20 D1 20 CE 20 B3 20 20 20 B6 DE 20 00 ; 0674
E390: 31 20 C0 DE 20 B2 20 20 20 C2 20 B2 20 B6 20 BB ; 0666
E3A0: 20 DA 20 CF 20 BD 20 A1 20 00 03 FF 12 16 14 FF ; 05E4

GAME OVER !
```

みゅ・う・じっ・く・…

音楽演奏

さて，いよいよ本書最後の挑戦である

マシン語による音楽演奏

に挑むことに致しましょう。

ＰＣ-8001は，基本的には，**ピー**という

BEEP音

しか出ません。しかし，本章のマスターにより，いろいろ複雑な音が出せるようになります。そして，その技法が実際に使えるよう，前章で学びました。

ＵＦＯの移動

の中に音楽を取り入れてみることにします。

さて，その音楽演奏ですが，さすがに一番最後にあるだけあって，前章までの各プログラムに比べやや複雑で，理解しにくいかもしれません。そこで，音楽演奏を**サブルーチン・パッケージ**の形で御紹介することに致します。ですから最初は，その**利用法**だけでもマスターするつもりで，気楽にお読みになってください。何度か読み返していただけば，そのうち理解できることでしょう。

ただし，——。

音楽演奏は，〝これ〟という決定的方法はありません。なぜならマシン語では，

音階を整数でとらえている

からです。もともと音階は，きっちりとした整数でとらえられる性質のものではありません。しかし，

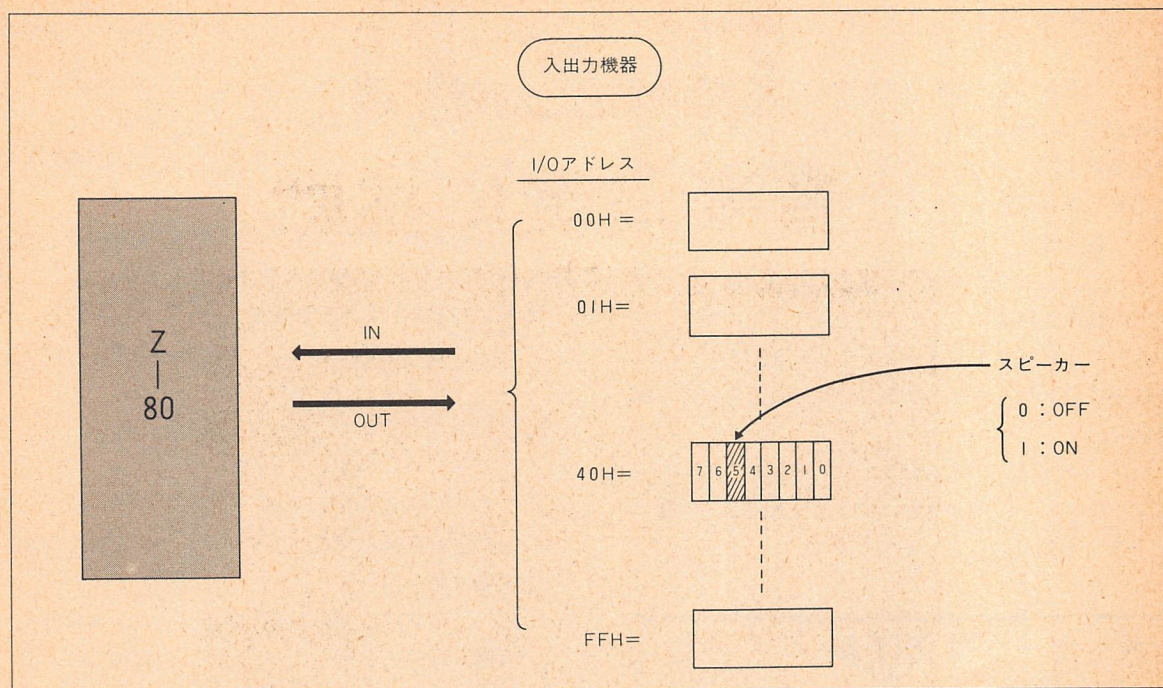
アリゴリズムのくふう

データの改良

等により，より美しい音楽に一歩でも近づくことが可能かもしれません。あなたも本章により，音楽演奏の基本的原理をマスターしていただけたら，その後は試行錯誤により，いろいろな方法を試みてください。より美しい**電子音**をかなでるために——。

スピーカーの制御

ところで，あなたのマシンのＣＰＵである**Ｚ—80**ですが，

256個までの入出力機器

第207図　スピーカーの制御

を制御することができます。各入出力機器には，アドレス（**I／Oアドレス**）

　　　　００H～ＦＦＨ

が割り振られています。そしてＺ―80の入出力命令

　　　　ＩＮ―――１バイトのデータを入力

　　　　ＯＵＴ――１バイトのデータを出力

で入出力機器とデータのやり取りができるのです。

さて音楽演奏の方法ですが，ＰＣ-8001に内蔵されているスピーカーを制御（音を出したり，止めたり）することを行います。スピーカーの制御ならBASICでも

　　　　BEEP　1

　　　　BEEP　0

で行うことができます。しかしながら，BASICでスピーカーを制御したのでは遅すぎて，ボケた音しか出せません。スピーカーの制御は速ければ速い程良いのです。

内蔵スピーカーを制御するには，ＯＵＴ命令で

　　　　Ｉ／Ｏアドレス４０Ｈのビット５

を制御することでできます。すなわち，

　　　　ビット５＝１……音を出す

　　　　ビット５＝０……音を止める

です（**第207図**）。

I/Oアドレス40H

ところでＩ／Ｏアドレス＝40Ｈですが，**ビット５以外の他のビットは**どうなっているのでしょう？　実は，これら他のビットもプリンター等，**他の機器の制御**のために使われています。

したがって，40Ｈを制御する際には，

　　　　ビット５以外をいじくってはダメ

なのです。ところが，**ＯＵＴ命令**を実行しますと，

　　　　１バイトのデータが出力される

　　　　⬇

　　　　４０Ｈの**すべてのビット**が制御される

ことになります。なんとか，４０Ｈのビット５だけを制御する方法はないものでしょうか？

実は，良い方法があります。と申しますのは有難いことに

N―BASICはＩ／Ｏアドレス４０Ｈ に出力した値を，いつもシステム・ワーク エリアのＥＡ６７Ｈにストアしている

という事実があるからです（その番地に**WOUT４０**と

いう名前をつけることにします)。

そこで,

LD A,(WOUT 4 0)

でいままで40Hに出力していた値を,Aレジスタに取り込みます。そしてZ―80のビット操作命令(〝Z―80活用表″を参照してください)を用い,

音を出す

ビット5=1……SET 5,A

音を止める

ビット5=0……RES 5,A

をほどこしてやります。こうして得られたAレジスタの値を

OUT (40H),A

でI/Oアドレスの40Hに出力してやれば,内蔵スピーカーを制御することができます。

メッセージは告げる

それでは,スピーカーをON,OFFするマシン語サブルーチンを具体的に作ってみることに致しましょう。

スピーカーを鳴らす

```
LD    A,(WOUT4 0) }
                  } データを作る
SET   5,A         }
OUT   (40H),A      ……鳴らす
```

スピーカーを止める

```
LD    A,(WOUT 4 0) }
                   } データを作る
RES   5,A          }
OUT   (40H),A       ……止める
```

これをアンセンブル・リストで示したのが,**第208**図です。

次にこれらのマシン語サブルーチンを実験するため,BASICでメイン・ルーチンを作ってみましょう。

まず,DEF USR関数を

USR0←―スピーカーを鳴らす

USR1←―スピーカーを止める

のように割り当てます。そして

U=USR0(0) ――――Ⓐ

U=USR1(1) ――――Ⓑ

のようにそれぞれのマシン語サブルーチンを呼び出せば良いのです。

(注) ここでは,マシン語サブルーチンと引数の受け渡しは行いませんから,( )の中は何でもかまいません。

ところで,この二つのサブルーチンを連続的に呼んだのでは,音がごく瞬間的に鳴るだけで,何が起こったかうまく確認できません。そこでⒶ,Ⓑの間に

FOR I=0 TO 2000:NEXT

のようにタイマーを入れてやります。さらに

PRINT "BEEP SWITCH ON"

PRINT "BEEP SWITCH OFF"

とコメントをつけてやれば,より一層ハッキリさせることができます。

```
                  ;=============================
                  ;  BEEP 1:BEEP2
                  ;    82.12.5:BY K.TSUKAGOSHI
                  ;=============================
                  ;
                          ORG   0C100H
                  ;
EA67              WOUT40:EQU   0EA67H
                  ;
C100 3A67EA       BEEP1:  LD    A,(WOUT40)     ;BEEP SWITCH ON
C103 CBEF                 SET   5,A
C105 D340                 OUT   (40H),A
C107 C9                   RET
                  ;
C108 3A67EA       BEEP0:  LD    A,(WOUT40)     ;BEEP SWITCH OFF
C10B CBAF                 RES   5,A
C10D D340                 OUT   (40H),A
C10F C9                   RET
                  ;
                          END
```

第208図 マシン語でスピーカーの制御

```
1000 '================================
1010 '   BEEP TEST
1020 '      1982.12.5:BY K.TSUKAGOSHI
1030 '================================
1040 '
1050 CLEAR 300,&HC7FF:DEFINT A-Z'              ;SET USR FUNCTION
1060 DEF USR0=&HC100'                          ;BEEP 1
1070 DEF USR1=&HC108'                          ;BEEP 0
1080 '
1085 PRINT
1090 PRINT "BEEP SWITCH ON ":U=USR0(0)'        ;BEEP SWITCH ON
1100   FOR I=0 TO 2000:NEXT'                   ;TIMER
1110 PRINT "BEEP SWITCH OFF":U=USR1(0)'        ;BEEP SWITCH OFF
1115 PRINT
1120 '
1130 END
```

第209図　BASICメイン・ルーチン(スピーカーの制御)

```
RUN                      ←── プログラムSTART

BEEP SWITCH ON           ←── と表示され，スピーカーが鳴る
```

第210図　スピーカーON

```
RUN

BEEP SWITCH ON
BEEP SWITCH OFF

OK
■
```

第211図　スピーカーOFF

こうしてまとめたBASICのメイン・ルーチンが，**第209図**です。以上の第208図，第209図のプログラムを入力しましたら，

RUN↘

でプログラムを走らせましょう。

最初は，**第210図**のように表示され，スピーカーが鳴ります。**USR0**が，うまく働いたわけです。やがてタイマーが切れ，**USR1**が働きますと，スピーカーの音が止まります（**第211図**）。

こうしてマシン語サブルーチンが，うまく働いたことが確認されました。まずは，**マシン語によるスピーカーの制御**が可能になったのです。

擬似サイン・カーブ

以上のスピーカーの制御法を基礎に，**音楽演奏の方法**を考えて行きます。

音――って何でしょう？

答――波です（**第212図**）。

これは，御存知ですね？　つまりマシン語を使い，スピーカーを制御し，第212図の状態を作り出してやれば，音楽を演奏することができるわけです。しかしながら，スピーカーは

ON：鳴る

OFF：止まる

の**二つの状態**しかとることはできません。第212図のようなサイン・カーブを描くことは，不可能です。

そこで，その代替として**第213図**のような波を考えます。少々荒っぽい波ですが，**この波ならスピーカーのON，OFFだけでも作れそう**です。すなわち

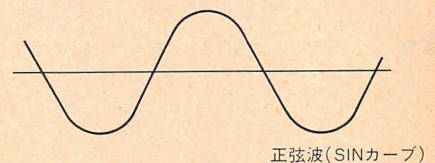
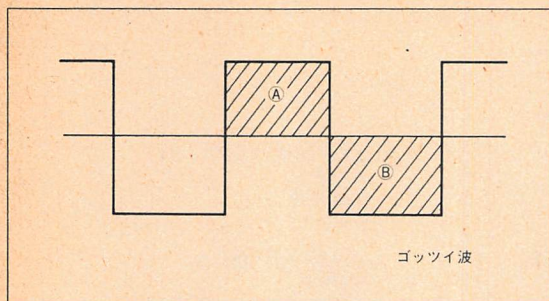
Ⓐ**の状態……スピーカーを鳴らす**

Ⓑ**の状態……スピーカーを止める**

のように制御してやれば良いのです。

それをマシン語で実現するには，次のように行えば良いでしょう。まず，

Bレジスタ←―適当な数

を入れておきます（これについては，後で説明します）。

第212図　音の正体

第213図　代替カーブ

これが，音の**半波長をはかる単位**になります。すると，第213図Ⓐの部分は

```
      LD    H, B             ……H←長さ
      LD    A, (WOUT40)   ┐
      SET   5, A          │音を出す
      OUT   (40H), A      ┘
  ㋑: DEC   H             ┐
      JR    NZ, ㋑        ┘長さ分ループ
```

で作れます。同様に音を止めるⒷの部分は，

```
      LD    H, B             ……H←長さ
      LD    A, (WOUT40)   ┐
      RES   5, A          │音を止める
      OUT   (40H), A      ┘
  ㋺: DEC   H             ┐
      JR    NZ, ㋺        ┘長さ分ループ
```

となります。こうして得られた**二組の半波長**を繰り返してやれば，音（擬似サイン・カーブ）を作り出すことができます。

音階の出し方

次に**音階**のつけ方です。

これも，サイン・カーブで見てみます。音の高低を調べてみますと，**第214図**のように

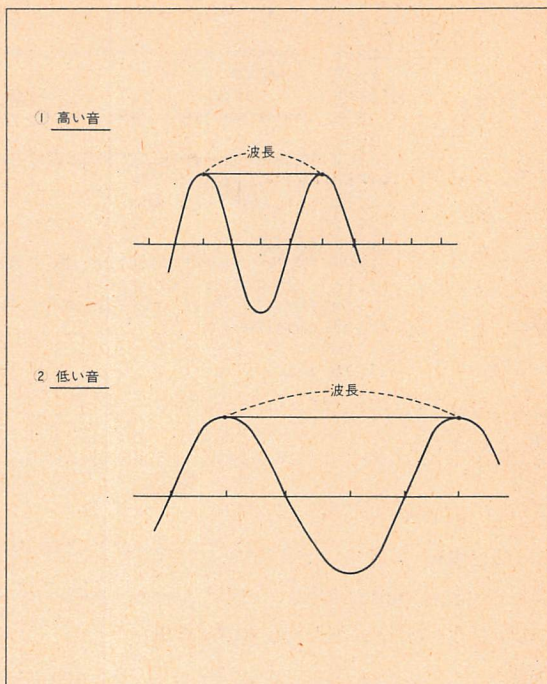
高い音……波長が短い

低い音……波長が長い

ことがわかります。**波長**というのは，**波の山から山までの距離**のことです。これは，感覚的にもおわかりになると思いますが，

細かい振動――――高い音

ゆったりした振動――低い音

のようになっているということです。

第214図　音の高低

音楽を奏でるには，**波の波長を変えてやれば良い**ことがわかりました。前節のプログラムで波長を変えるには，いかが致したらよろしいでしょうか？

簡単です。Bレジスタに入れた**適当な数の大きさ**を変えてやれば良いのです。プログラム本体に手を加える必要はありません。

歪んだ音色

次が，**音符の長さ**です。♩とか♩とか，♪とかの区別はどのようにしたら良いのでしょうか？

それには，まず**基準にする音符**も決めます。基準は，

もっとも短い音（その曲の中で）

を選びます。たとえば，

基準：８分音符＝♪

を選んだとします（**第215図**）。そして，その基準となる音符の長さを

DEレジスタをカウンタ

に使ってはかります。DEの値が大きい程，基準の長さが長くなります。また，アップ・テンポの曲でしたらDEに入れる値を小さくします。

具体的にどうやるかを，図で示します。

もっともわかりやすいのは，**第216図**の方法でしょう。

まずこれで1波長分の音が出ます。続いて，DEレジスタから1を引きます。0になっていなければ，さらに1波長演奏します。こうして，

DEレジスタ＝0

になったら基準の長さがおわり，というわけです。

第215図　音の長さは？

実は，この方法はわかりやすいのですが，実際に実験してみますと，調子っぱずれになるのがわかります。なぜでしょう？　もちろん理由があります。

第216図の右側を御覧になってください。

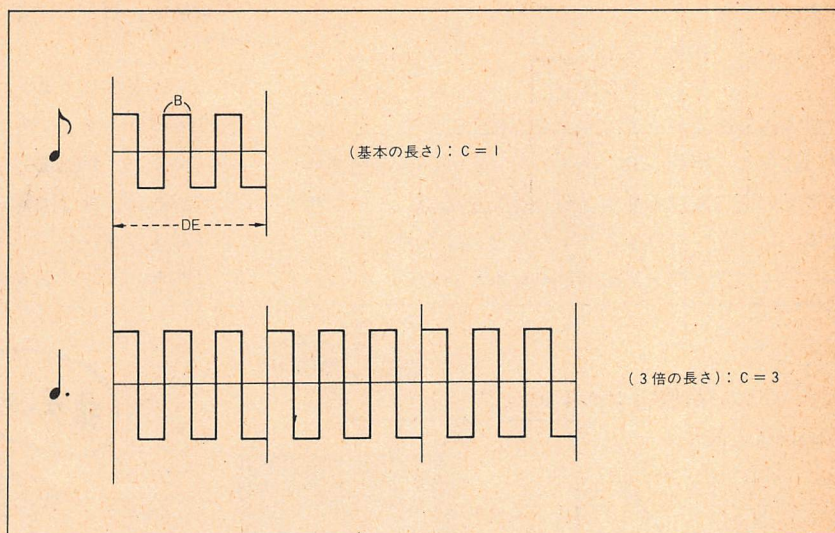
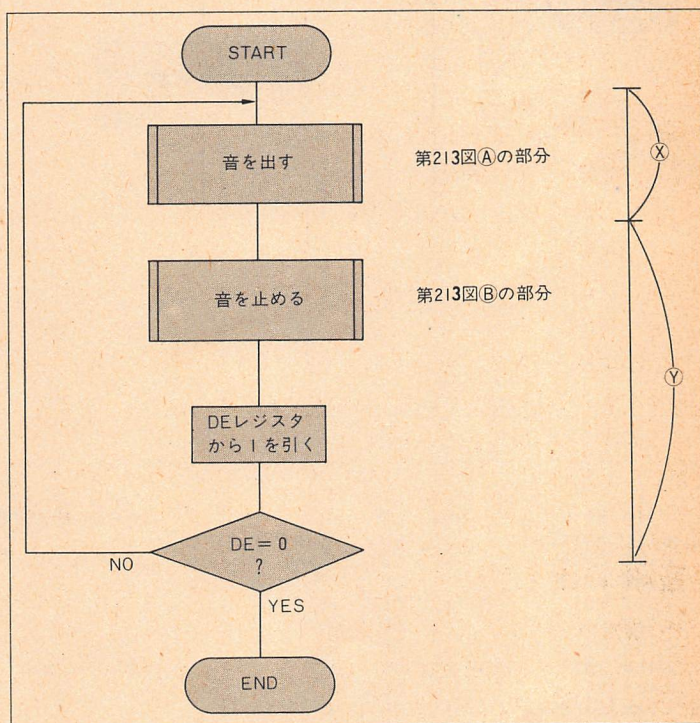
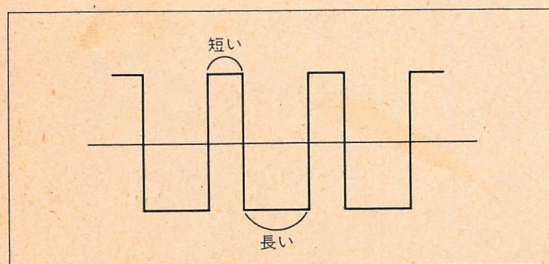
Ⓧ　　　　　Ⓨ
‖　　　　　‖
（音が出ている時間）＜（音が止まっている時間）

のようになっていますね。これは，

DEレジスタから1を引く
DE＝0　かを判定する

間も音が止まっているため，音の止まっている時間が短くなるからです。したがってこのアルゴリズムで作られた波は，**第217図**のように

歪んだ波

になってしまうのです。これでは，美しい調べを奏でることはできませんね？

第216図　基準の音を出す（その1）

第217図　ゆがんだ波

基本1音符の出力

やはり音楽らしく聞かせるには，

音の出ている時間＝音の止まっている時間

にする必要があります。それには，

DEレジスタから1を引く
DEレジスタの値＝0

のチェックを

　　音を出す部分

　　音を止める部分

に組み込む必要があります（**第218図**）。すると，ループの途中**二ヵ所**から飛び出すことになり，構造化プログラミングにおける基本三構造違反を犯すことになります。しかし，これもきれいな音を出すため

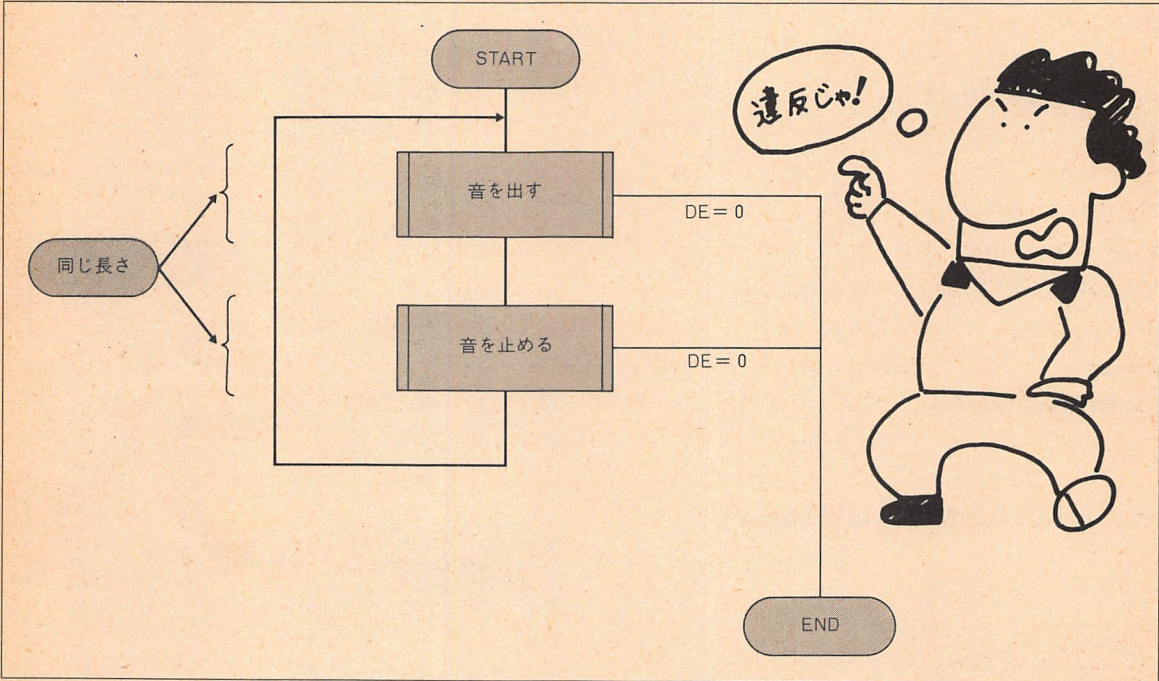
　　ソフトウェアの割込みがかかった

と解釈し，目をつぶることに致しましょう。

以上の考察で，基本の長さの音（第215図の上図）を出すサブルーチンが作れます。名前を

PLSB

とつければ，**第219図**のようになります。簡単に説明しておきます。

このサブルーチンを呼ぶ前に

第218図　バランス良く

```
C15A 60       PLSB:  LD   H,B          ;H=SCALE CODE
C15B 3A67EA          LD   A,(WOUT40)
C15E CBEF            SET  5,A          ;BEEP 1
C160 D340            OUT  (40H),A
C162 1B       PL1:   DEC  DE
C163 7A              LD   A,D
C164 B3              OR   E
C165 2815            JR   Z,PL3        ;LENGTH END
C167 25              DEC  H
C168 20F8            JR   NZ,PL1
C16A 60              LD   H,B
C16B 3A67EA          LD   A,(WOUT40)
C16E CBAF            RES  5,A          ;BEEP 0
C170 D340            OUT  (40H),A
C172 1B       PL2:   DEC  DE
C173 7A              LD   A,D
C174 B3              OR   E
C175 2805            JR   Z,PL3
C177 25              DEC  H
C178 20F8            JR   NZ,PL2
C17A 18DE            JR   PLSB
C17C C9       PL3:   RET
```


第219図　基本1音符の出力

B←半波長の長さ } 第215図の上図参照
DE←基本1音符の長さ }

のようにセットしておきます。

PLSB：LD　H，B
　　　　⁝
　　　JR　NZ，PL1

の部分が前にやりましたように音を出すところです。その中にある

DEC　DE
LD　A，D
OR　E

がDE＝0チェックをしているところです。ここでは，**論理演算OR**を用いてチェックしていますが，あなたのわかりやすい方法でチェックすれば良いでしょう。音を止める部分も同様です。

以上が基本となる音符を出力するサブルーチンです。音楽演奏全体の中では，この部分が一番理解しにくいかもしれません。ともかく，先の値をセットして

CALL　PLSB

とすると，**音が基本の長さだけ出力される**と理解してください。

長音の演奏は？

サブルーチンPLSBができれば，その二倍，三倍の長さの音を出すのは簡単です。**Cレジスタをカウンタ**に使い，基本の長さの倍数だけPLSBをCALLしてやれば良いのです。たとえば三倍の長さの音を出したいなら，Cの値を03Hにしておき

Ⓐ：PUSH　DE
　　CALL　PLSB
　　POP　DE
　　DEC　C
　　JR　NZ，Ⓐ

のようにすれば良いのです。ここで**DEレジスタの値を保存**しているのは，PLSBをCALLするたびに

DE＝0

となってしまうからです。

以上の部分をアセンブル・リストで示したのが，**第220図**です。名前を**PLAY**としてあります。このサブルーチンは，

B←音階コード（半波長のカウンタ）
C←基本音符の何倍の長さかを示す値
DE←基本1音符の長さ

を入れてCALLすると，希望の音符が演奏されるというものです。

さて，PLAYの中で**HLレジスタ**が出てきているのに御注意ください。このHL値は，PLAYの中で

PUSH，POP命令

により保存されています。そして最後に，HLレジスタの値が一つ大きくなっています。このHLレジスタの働きがわかれば，音楽演奏全体のサブルーチンができ上がります。

```
C14E E5        PLAY:  PUSH HL          ;HL=POINTER
C14F D5        PY1:   PUSH DE          ;DATA OF LENGTH
C150 CD5AC1           CALL PLSB        ;PLAY SUB
C153 D1               POP  DE
C154 0D               DEC  C
C155 20F8             JR   NZ,PY1
C157 E1               POP  HL          ;HL=POINTER
C158 23               INC  HL
C159 C9               RET
```

第220図　サブルーチンPLAY

音楽演奏サブルーチンの完成

さて，以上まででサブルーチンPLAYが完成しました。しかし，これだけでは

音符一つ

しか演奏ができません。音符一つでは，音楽になりませんね？

そこでたくさんの音符を連続して演奏することを考えてみます。それには，音符のデータを**データ列**として用意すれば良いのです。1音符のデータは，

音階データ（半波長）
長さ（基本音符の何倍かを表わす）

の二組で構成されます。したがって演奏したい曲を，

1音符＝2データ

の割で**データ化**して行き，**音符のデータ列**を作ります。

演奏部分全体のサブルーチンを**SONG**という名前にすれば，SONGは次のようになります。

LD　B，(HL)
　　……音階データ

```
C100 7E        SONG:  LD   A,(HL)        ;IN(HL=MUSIC DATA)  } 演奏終了のチェック
C101 A7               AND  A
C102 C8               RET  Z
C103 47               LD   B,A           ;B=SCALE CODE       } 音符データの取り込み
C104 23               INC  HL
C105 4E               LD   C,(HL)        ;C=LENGTH
C106 07               RLCA               ;bit7=1(PAUSE) ?    } 休止符？
C107 3005             JR   NC,SO1
C109 CD21C1           CALL PAUSE                             } 休止符演奏
C10C 1803             JR   SO2
C10E CD4EC1    SO1:   CALL PLAY                              ── 音符の演奏
C111 3A67EA    SO2:   LD   A,(WOUT40)    ;BEEP 0             } 音を止める
C114 CBAF             RES  5,A
C116 D340             OUT  (40H),A
C118 0610             LD   B,10H
C11A CD7DC1    SO3:   CALL TIME                              } 区切り用タイマー
C11D 10FB             DJNZ SO3                               ── 次の音符へ
C11F 18DF             JR   SONG
```

第221図　SONGの完成

　　INC　HL
　　LD　　C,（HL）　……長さ
　　CALL　PLAY

これで1音符分の音が出ます。先に見ましたように，PLAYをCALLすると，HLレジスタが1つ大きくなります。ということは，

HLレジスタは次の音符データの先頭

を指すことになりますから，

　　JR　　SONG

でループしてやれば，演奏を続けることになります。サブルーチンPLAYの中でHLレジスタの値を1つ進めていたのは，このためだったのです。

　ここで注意しなければならないことは，このままでは，いつまでたっても**演奏がSTOPしない**ということです。そこで，データ列の最後に

ENDマーク＝00H

を設けてやり，データとして00Hを拾ったら演奏を中止するように組んでやります。

　こうしてでき上がった音楽全体のサブルーチンが，**第221図**です。次節でこの**SONG**の説明を致します。

SONGの解析

① 演奏終了のチェック

　　LD　　A,（HL）
　　AND　A
　　RET　Z

　音符データが0かどうかを調べ，もし0ならENDマークなので演奏をやめます。

② 音符データの取り込み

　　LD　　B,A　　……音階
　　INC　HL
　　LD　　C,（HL）　……長さ

　Bレジスタ，Cレジスタに二組のデータを取り込みます。

③ 休止符チェック

　　RLCA
　　JR　　NC,SO1

　音楽演奏の場合，**休止符も考慮**しなければなりません。音符であるか，休止符であるかをどのようにチェックしても良いのですが，ここでは

　　音階データの7ビット目

で区別するように組んでみました。すなわち

　　ビット7＝｛0：普通の音符／1：休止符｝

のように区別しています。ですから休止符を使いたい時は，

　　｛音階データ＝FFH（ビット7＝1）／長さデータ＝休止したい長さを指定｝

このようにデータを用意すれば良いのです。

　プログラムの中では，

　　RLCA

でビット7の値をキャリー・フラグに取り込み，判定しています。

④ 休止符の演奏

　　CALL　PAUSE
　　JR　　SO2

```
                    ;============================
                    ;  MUSIC SUB
                    ;    82.12.5:BY K.TSUKAGOSHI
                    ;============================
                    ;
                              ORG   0C100H
                    ;
EA67                WOUT40:EQU   0EA67H
                    ;
C100 7E             SONG:    LD    A,(HL)          ;IN(HL=MUSIC DATA)
C101 A7                      AND   A
C102 C8                      RET   Z
C103 47                      LD    B,A             ;B=SCALE CODE
C104 23                      INC   HL
C105 4E                      LD    C,(HL)          ;C=LENGTH
C106 07                      RLCA                  ;bit7=1(PAUSE) ?
C107 3005                    JR    NC,SO1
C109 CD21C1                  CALL  PAUSE
C10C 1803                    JR    SO2
C10E CD4EC1         SO1:     CALL  PLAY
C111 3A67EA         SO2:     LD    A,(WOUT40)      ;BEEP 0
C114 CBAF                    RES   5,A
C116 D340                    OUT   (40H),A
C118 0610                    LD    B,10H
C11A CD7DC1         SO3:     CALL  TIME
C11D 10FB                    DJNZ  SO3
C11F 18DF                    JR    SONG
                    ;
C121 E5             PAUSE:   PUSH  HL
C122 D5             PS1:     PUSH  DE
C123 CD2DC1                  CALL  PASB            ;PAUSE SUB
C126 D1                      POP   DE
C127 0D                      DEC   C
C128 20F8                    JR    NZ,PS1
C12A E1                      POP   HL
C12B 23                      INC   HL
C12C C9                      RET
                    ;
C12D 60             PASB:    LD    H,B
C12E 3A67EA                  LD    A,(WOUT40)
C131 CBAF                    RES   5,A
C133 D340                    OUT   (40H),A
C135 1B             PA1:     DEC   DE
C136 7A                      LD    A,D
C137 B3                      OR    E
C138 2813                    JR    Z,PA3
C13A 25                      DEC   H
C13B 20F0                    JR    NZ,PASB
C13D 60                      LD    H,B
C13E 3A67EA                  LD    A,(WOUT40)
C141 CBAF                    RES   5,A
C143 1B             PA2:     DEC   DE
C144 7A                      LD    A,D
C145 B3                      OR    E
C146 2805                    JR    Z,PA3
C148 25                      DEC   H
C149 20F8                    JR    NZ,PA2
C14B 18E0                    JR    PASB
C14D C9             PA3:     RET
                    ;
C14E E5             PLAY:    PUSH  HL              ;HL=POINTER
C14F D5             PY1:     PUSH  DE              ;DATA OF LENGTH
C150 CD5AC1                  CALL  PLSB            ;PLAY SUB
C153 D1                      POP   DE
C154 0D                      DEC   C
C155 20F8                    JR    NZ,PY1
C157 E1                      POP   HL              ;HL=POINTER
C158 23                      INC   HL
C159 C9                      RET
```

〈次頁に続く〉

```
                    ;
C15A 60       PLSB:  LD    H,B           ;H=SCALE CODE
C15B 3A67EA          LD    A,(WOUT40)
C15E CBEF            SET   5,A           ;BEEP 1
C160 D340            OUT   (40H),A
C162 1B       PL1:   DEC   DE
C163 7A              LD    A,D
C164 B3              OR    E
C165 2815            JR    Z,PL3         ;LENGTH END
C167 25              DEC   H
C168 20F8            JR    NZ,PL1
C16A 60              LD    H,B
C16B 3A67EA          LD    A,(WOUT40)
C16E CBAF            RES   5,A           ;BEEP 0
C170 D340            OUT   (40H),A
C172 1B       PL2:   DEC   DE
C173 7A              LD    A,D
C174 B3              OR    E
C175 2805            JR    Z,PL3
C177 25              DEC   H
C178 20F8            JR    NZ,PL2
C17A 18DE            JR    PLSB
C17C C9       PL3:   RET
                    ;
C17D C5       TIME:  PUSH  BC
C17E 0E00            LD    C,0
C180 0D       TI1:   DEC   C
C181 20FD            JR    NZ,TI1
C183 C1              POP   BC
C184 C9              RET
                    ;
                     END
```

第222図　音楽演奏サブルーチンの完成

PAUSEは，休止符を演奏するサブルーチンです。我々は，まだこのサブルーチンを作っていませんでした。**第222図**を御覧ください。これは，今まで作ってきたサブルーチンを集め，音楽演奏ルーチンにまとめたものです。いわば完成版です。あとはデータを集めるだけで音楽演奏が可能です。

この中に休止符用のPAUSEが見られます。このルーチンを目で追ってみてくだたい。**PLAYとまったく同じ**であることが，おわかりになるでしょう。ただ，

　　ＳＥＴ命令──→ＲＥＳ命令

に変えて，音を出していないだけです。

⑤　**（音の出る）音符の演奏**

```
        CALL  PLAY
```

⑥　**音の停止**

```
        LD    A, (WOUT40)
        RES   5, A
        OUT   (40H), A
```

音を止めます。この処理は不要に見えますが，ここで音を止めておかないと，**次に休止符が来ても音が鳴り続ける**ことがあります。

⑦　**タイマー**

```
        LD    B, 10
SO3:    CALL  TIME
        DJNZ  SO3
```

これは，音符と音符の間に区切りを入れるためで，無くてもかまいません。

⑧　**次の音へ**

```
        JR    SONG
```

1オクターブとは

最後に残りましたのが，

音階データの計算

です。これさえわかれば，すでにサブルーチンはできあがっていますので，すぐにでも演奏可能です。

音階データを計算するには，やはり**音楽の知識**が必要です。その基本は，次のとおりです。

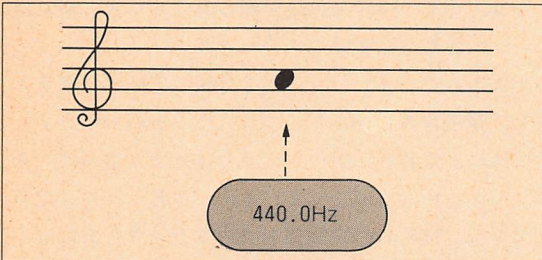
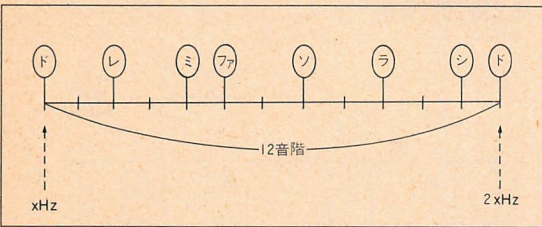
①　**基準となる音（ハ長調のラ）**の周波数は，

440.0Hz（ヘルツ）

である。
② **1オクターブ**上がると，**周波数**は**2倍**になる。
③ 1オクターブは，**12音階**である。

以上の関係を図示したのが，**第223図，第224図**です。この規則を満たすように，

音階データを計算

するわけです。たとえば基準のラの音を出すには，音階データをいくつにすれば良いでしょうか？

第223図　基準の音

第224図　1オクターブ

それには，Z—80の各命令の**実行時間**を知る必要があります。命令の実行時間は，

ステート（state）

という単位ではかります。ステートは，付録の〝インストラクション一覧表〟にのっています。たとえば

LD　　A，B　……4ステート
ADD　A，5　……7ステート

といった具合に時間がかかります。

それでは，**1ステートはどの位の時間**でしょう？ステートは，各命令を実行する際の基本となる

周　期　（単位：秒）

です。周期については，次の有名な公式があります。

$$T=\frac{1}{f}\quad\left(\begin{array}{l}T：周　期\\ f：周波数\end{array}\right)$$

PC-8001のfは，「ユーザーズ・マニュアル」のP.72にのっています。

$$3.9936\text{MHz}=3.9936\times10^{6}\ \text{Hz}$$

です。これを先の公式に代入しますと，ステートが求まります。

$$ステート=\frac{1}{3.9936\times10^{6}}=0.2504\times10^{-6}\ (秒)=0.2504\ (\mu秒)$$

と計算されます。これが，PC-8001の1ステートの長さです。

音階データの計算

1ステートに要する時間が出ました。そこで第222図のPLSBルーチン中で，

1波長を作っている命令群
の全ステート数

を数えます。それが，Tステートメントであったとします。そして，**基準の音ラ**を奏でる音階データをxとすれば，

$$0.2504\times10^{6}Tx\ (秒)$$

が，ラの音の**周期**ということになります。周波数を求めるには，先の公式からこの逆数を求めれば良く，その値が440.0Hzとわかっていますから，

$$\frac{1}{0.2504\times10^{6}Tx}=440.0$$

$$\therefore x=\frac{1}{0.2504\times10^{6}T\times440.0}$$

Tの値は，ゆっくり数えれば出ますからその値をこの式に入れて計算すれば，xの値が求まります。

こうして，他の音階についても次々と計算できるのですが，ここに一つ問題があります。というのは，ここで計算に用いた各命令のステートの数え方は，〝Z—80（PC-8001では，Z—80に相当するNECのμPD780C—1）のそれを用いたということです。ということは，これはZ—80の性能に負荷をかけず，**フル性能の状態で使った場合の計算**です。実際のPC-8001のステート数は，残念ながらもっと遅いのです。

それなら，PC-8001における各命令の

正しいステート数はいくつ？

かといえば，残念ながらメーカーからは発表されてい

ませんのでわかりません。結論を申し上げますと，

ＰＣ－8001では，正しい音階データ
の計算はできない

ということです。

ハーモニーは苦手

正確ではないにしろ，前節の計算法にしたがって得られた音階データで演奏をすると，どういうことになるでしょうか？

答――一応，音楽らしく聞こえます。ただし，周波数がズレていますから**他の楽器とのハーモニーは不可能**です。結局，ＰＣ－8001での音楽演奏は，

ＰＣ－8001単体で使う

しかないようです。それでも，ＧＡＭＥ等の効果音に使うには十分ですから御安心を。

(注)　耳の良い人は，いろいろなデータで音を出して聞き比べてください。正しい基準の音のデータが見つかるかもしれません。基準の音のデータさえ見つかれば，他の音は計算で求まります。

さて，どうせ正しいデータでないなら，

1オクターブ上がれば周波数は2倍
1オクターブを12音階に分ける　}―(※)

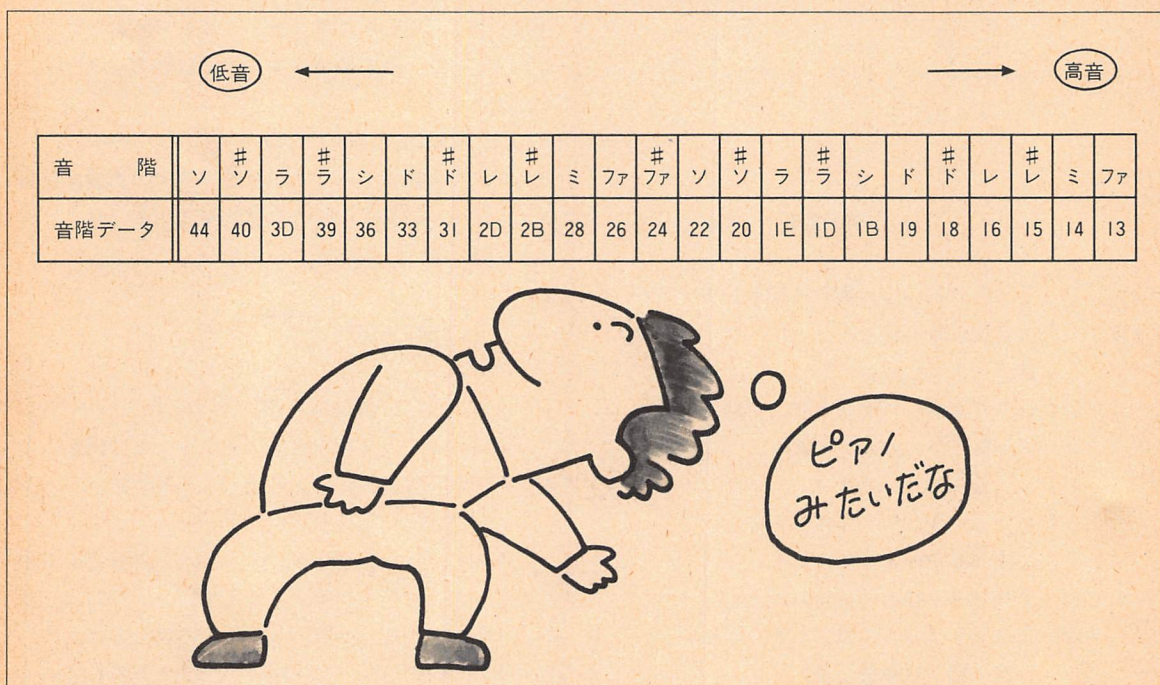
の二条件を満たすデータなら，何でも良いことになります。それなら，何も自分で音階データなんて計算する必要はありませんね？

かつてＰＣ－8001の御先祖様にあたるＴＫ―80では，

各命令の正確なステート数

が発表されていました。そして付属の応用プログラムにも，**〝電子オルゴール〞**のプログラムがのっていました。他にもＴＫ―80関係の文献を捜せば，音階データが見つかるかもしれません。また，ＴＫ―80にこだわらずとも，何か(※)の関係を満たすデータがあればそれが使えます。

第225図に音階データの一例を紹介しておきます。

音階	ソ	♯ソ	ラ	♯ラ	シ	ド	♯ド	レ	♯レ	ミ	ファ	♯ファ	ソ	♯ソ	ラ	♯ラ	シ	ド	♯ド	レ	♯レ	ミ	ファ
音階データ	44	40	3D	39	36	33	31	2D	2B	28	26	24	22	20	1E	1D	1B	19	18	16	15	14	13

第225図　音階データの一例

UFO用音楽の定義

さて，長らく長くお待たせ致しました。音楽演奏サブルーチンSONGは，完成しています。また音階データも揃いました。いよいよ，

音楽演奏の実験

にとりかかる時がやってきました。前章でやりました

ＵＦＯの移動

に音楽をつけてみましょう。これで

音楽付きの

カラー・グラフィック

が可能になります。

そこで使いますのが，**第226図**の曲です。何と，(大きい声ではいえませんが) あのチャルメラです。あ～，何たる野暮ったいことよ。あんなもの曲か！ とマア，そう堅いことは言わないで――。こちらあたりからボチボチ入るのが，良いのとちゃいまっしゃろか？

第226図　ＵＦＯで使う音楽

そこで，この楽譜に合わせて音楽のデータ列を作ります。まず**音階コードの変換から**。

ド＝３３Ｈ

レ＝２ＤＨ

ミ＝２８Ｈ

レ＝２ＤＨ

～

のように第225図を見ながら変換して行きます。

次に**長さ**です。

この曲の中で一番短い音符は，♪（八分音符）です。これを基本の長さ１に取りましょう。すると，

ド（長さ１）＝３３Ｈ，０１Ｈ

レ（長さ１）＝２ＤＨ，０１Ｈ

ミ（長さ３）＝２８Ｈ，０３Ｈ

レ（長さ１）＝２ＤＨ，０１Ｈ

～

のように変換されます。

以上のようにして得られたデータを，データ列（名前を**ＣＨＡＬ**とでもつけましょう）として

CHAL：ＤＢ　～

で定義しておけば良いのです。

```
                  ;==============================
                  ;  MUSIC (チャルメラ)
                  ;     82.12.5:BY K.TSUKAGOSHI
                  ;==============================
                  ;
                           ORG   0C100H
                  ;
5C66              MON:     EQU   5C66H
EA67              WOUT40:EQU    0EA67H
                  ;
C100 11FF15       USR:     LD    DE,15FFH
C103 210CC1                LD    HL,CHAL
C106 CD23C1                CALL  SONG
C109 C3665C                JP    MON
                  ;
C10C 33012D01     CHAL:    DB    33H,1,2DH,1,28H,3,2DH,1,33H,2
C110 28032D01
C114 3302
C116 33012D01              DB    33H,1,2DH,1,28H,1,2DH,1,33H,1
C11A 28012D01
C11E 3301
C120 2D0500                DB    2DH,5,0
                  ;
C123 7E           SONG:    LD    A,(HL)         ;IN(HL=MUSIC DATA)
C124 A7                    AND   A
C125 C8                    RET   Z
C126 47                    LD    B,A            ;B=SCALE CODE
C127 23                    INC   HL
C128 4E                    LD    C,(HL)         ;C=LENGTH
C129 07                    RLCA                 ;bit7=1(PAUSE) ?
C12A 3005                  JR    NC,S01
C12C CD44C1                CALL  PAUSE
C12F 1803                  JR    S02
C131 CD71C1       S01:     CALL  PLAY
C134 3A67EA       S02:     LD    A,(WOUT40)     ;BEEP 0
C137 CBAF                  RES   5,A
C139 D340                  OUT   (40H),A
C13B 0610                  LD    B,10H
C13D CDA0C1       S03:     CALL  TIME
C140 10FB                  DJNZ  S03
```

〈次頁に続く〉

```
C142 18DF          JR    SONG
                ;
C144 E5         PAUSE: PUSH HL
C145 D5         PS1:   PUSH DE
C146 CD50C1            CALL PASB          ;PAUSE SUB
C149 D1                POP  DE
C14A 0D                DEC  C
C14B 20F8              JR   NZ,PS1
C14D E1                POP  HL
C14E 23                INC  HL
C14F C9                RET
                ;
C150 60         PASB:  LD   H,B
C151 3A67EA            LD   A,(WOUT40)
C154 CBAF              RES  5,A
C156 D340              OUT  (40H),A
C158 1B         PA1:   DEC  DE
C159 7A                LD   A,D
C15A B3                OR   E
C15B 2813              JR   Z,PA3
C15D 25                DEC  H
C15E 20F0              JR   NZ,PASB
C160 60                LD   H,B
C161 3A67EA            LD   A,(WOUT40)
C164 CBAF              RES  5,A
C166 1B         PA2:   DEC  DE
C167 7A                LD   A,D
C168 B3                OR   E
C169 2805              JR   Z,PA3
C16B 25                DEC  H
C16C 20F8              JR   NZ,PA2
C16E 18E0              JR   PASB
C170 C9         PA3:   RET
                ;
C171 E5         PLAY:  PUSH HL            ;HL=POINTER
C172 D5         PY1:   PUSH DE            ;DATA OF LENGTH
C173 CD7DC1            CALL PLSB          ;PLAY SUB
C176 D1                POP  DE
C177 0D                DEC  C
C178 20F8              JR   NZ,PY1
C17A E1                POP  HL            ;HL=POINTER
C17B 23                INC  HL
C17C C9                RET
                ;
C17D 60         PLSB:  LD   H,B           ;H=SCALE CODE
C17E 3A67EA            LD   A,(WOUT40)
C181 CBEF              SET  5,A           ;BEEP 1
C183 D340              OUT  (40H),A
C185 1B         PL1:   DEC  DE
C186 7A                LD   A,D
C187 B3                OR   E
C188 2815              JR   Z,PL3         ;LENGTH END
C18A 25                DEC  H
C18B 20F8              JR   NZ,PL1
C18D 60                LD   H,B
C18E 3A67EA            LD   A,(WOUT40)
C191 CBAF              RES  5,A           ;BEEP 0
C193 D340              OUT  (40H),A
C195 1B         PL2:   DEC  DE
C196 7A                LD   A,D
C197 B3                OR   E
C198 2805              JR   Z,PL3
C19A 25                DEC  H
C19B 20F8              JR   NZ,PL2
C19D 18DE              JR   PLSB
C19F C9         PL3:   RET
                ;
C1A0 C5         TIME:  PUSH BC
C1A1 0E00              LD   C,0
C1A3 0D         TI1:   DEC  C
C1A4 20FD              JR   NZ,TI1
C1A6 C1                POP  BC
C1A7 C9                RET
                ;
                       END
```

第227図　チャルメラに挑戦

あのチャルメラの音が

データが準備されましたら、演奏は次のようにやれば良いのです。

```
LD    DE, 15FFH ——①
LD    HL, CHAL  ——②
CALL  SONG      ——③
JP    MON       ——④
```

① 基本音符（ここでは八分音符）の長さを決めます。この数により，音楽全体のスピードが変わります。
② 音楽のデータ列の先頭をセットします。
③ 音楽演奏ルーチンをCALLします。
④ マシン語モニタへ。

こうしてできあがりました，チャルメラ演奏プログラム（なげかわしい！）が，**第227図**です。大変，大変お待たせ致しました。走らせますよ。

GC100↘

お〜〜〜！　あのなつかしい音が！　感激！感激！
あー，なぜかラーメンが食べたくなってきましたね。

UFOのチャルメラ屋さん

このチャルメラを，ＵＦＯの中に取り入れます（何という奇妙キテレツなる取り合わせ！）。仕掛けとしましては，

ＵＦＯが方向転換

する時に，チャルメラの音が聞こえるように致しましょう。すると，ＵＦＯのチャルメラ屋さんが出現したことになります，ハイ。新商売の時代ですね。？

まずは，マシン語サブルーチン。

チャルメラ（第227図）のＣ１０９Ｈの

ＪＰ　　ＭＯＮ

を，

ＲＥＴ

に代えます。BASIC のメイン・ルーチンから呼ばれた時に，またBASIC に帰るためです。この変更を加えた上で，チャルメラ演奏ルーチンを第204図の後にくっつけてひきます。これでマシン語サブルーチンができあがりました（**第228図**）。

次が，メイン・ルーチンです。

これも第205図に少し手を加えるだけででき上がります。まず，**ユーザー関数**が１つ増えます。

```
                    ;=================
                    ;  PRINT UFO-3
                    ;     (82.12.3)
                    ;=================
                    ;
                            ORG   0C800H
                    ;
03F3                LOC:    EQU   3F3H          ;LOCATE TO ADDRESS
EA67                WOUT40:EQU    0EA67H
                    ;
C800 5E             PUFO:   LD    E,(HL)        ;PRINT UFO FROM BASIC
C801 1C                     INC   E             ;(1,1) ORIGIN
C802 23                     INC   HL
C803 56                     LD    D,(HL)        ;DE=LOCATE OF UFO
C804 14                     INC   D
C805 EB                     EX    DE,HL
C806 CDF303                 CALL  LOC
C809 113AC8                 LD    DE,DUFO       ;UFO DATA
C80C CD0FC8                 CALL  PU1
                    ;
C80F E5             PU1:    PUSH  HL            ;IN(DE=DATA,HL=ADDRESS)
C810 0609                   LD    B,9           ;9 BYTES PER 1 LINE
C812 1A             PU2:    LD    A,(DE)
C813 77                     LD    (HL),A
C814 13                     INC   DE
C815 23                     INC   HL
C816 10FA                   DJNZ  PU2
C818 E1                     POP   HL
C819 017800                 LD    BC,120        ;HL=HL+120
C81C 09                     ADD   HL,BC
C81D C9                     RET
                    ;
C81E 5E             EUFO:   LD    E,(HL)        ;ERASE UFO
```

〈次頁に続く〉

```
C81F 1C                 INC  E
C820 23                 INC  HL
C821 56                 LD   D,(HL)
C822 14                 INC  D
C823 EB                 EX   DE,HL          ;HL=LOCATE OF ERASE UFO
C824 CDF303             CALL LOC
C827 AF                 XOR  A              ;FOR ERASE CODE
C828 0E02               LD   C,2            ;2 LINES
C82A 117800             LD   DE,120         ;FOR HL=HL+120
C82D E5        EU1:     PUSH HL
C82E 0609               LD   B,9
C830 77        EU2:     LD   (HL),A
C831 23                 INC  HL
C832 10FC               DJNZ EU2
C834 E1                 POP  HL
C835 19                 ADD  HL,DE
C836 0D                 DEC  C
C837 20F4               JR   NZ,EU1
C839 C9                 RET
               ;
C83A 80ECEEBF  DUFO:    DB   80H,0ECH,0EEH,0BFH,99H,0FBH,0EEH,0CEH,8
C83E 99FBEECE
C842 08
C843 33E67F22           DB   33H,0E6H,7FH,22H,77H,22H,0F7H,6EH,33H
C847 7722F76E
C84B 33
               ;
C84C 11FF15    SOUND:   LD   DE,15FFH       ;ﾁｬﾙﾒﾗ
C84F 2156C8             LD   HL,CHAL
C852 CD6DC8             CALL SONG
C855 C9                 RET
               ;
C856 33012D01  CHAL:    DB   33H,1,2DH,1,28H,3,2DH,1,33H,2
C85A 28032D01
C85E 3302
C860 33012D01           DB   33H,1,2DH,1,28H,1,2DH,1,33H,1
C864 28012D01
C868 3301
C86A 2D0500             DB   2DH,5,0
               ;
C86D 7E        SONG:    LD   A,(HL)         ;IN(HL=MUSIC DATA)
C86E A7                 AND  A
C86F C8                 RET  Z
C870 47                 LD   B,A            ;B=SCALE CODE
C871 23                 INC  HL
C872 4E                 LD   C,(HL)         ;C=LENGTH
C873 07                 RLCA                ;bit7=1(PAUSE) ?
C874 3005               JR   NC,SO1
C876 CD8EC8             CALL PAUSE
C879 1803               JR   SO2
C87B CDBBC8    SO1:     CALL PLAY
C87E 3A67EA    SO2:     LD   A,(WOUT40)     ;BEEP 0
C881 CBAF               RES  5,A
C883 D340               OUT  (40H),A
C885 0610               LD   B,10H
C887 CDEAC8    SO3:     CALL TIME
C88A 10FB               DJNZ SO3
C88C 18DF               JR   SONG
               ;
C88E E5        PAUSE:   PUSH HL
C88F D5        PS1:     PUSH DE
C890 CD9AC8             CALL PASB           ;PAUSE SUB
C893 D1                 POP  DE
C894 0D                 DEC  C
C895 20F8               JR   NZ,PS1
C897 E1                 POP  HL
C898 23                 INC  HL
C899 C9                 RET
               ;
C89A 60        PASB:    LD   H,B
C89B 3A67EA             LD   A,(WOUT40)
C89E CBAF               RES  5,A
C8A0 D340               OUT  (40H),A
C8A2 1B        PA1:     DEC  DE
C8A3 7A                 LD   A,D
C8A4 B3                 OR   E
C8A5 2813               JR   Z,PA3
```

```
C8A7 25                 DEC  H
C8A8 20F0               JR   NZ,PASB
C8AA 60                 LD   H,B
C8AB 3A67EA             LD   A,(WOUT40)
C8AE CBAF               RES  5,A
C8B0 1B        PA2:     DEC  DE
C8B1 7A                 LD   A,D
C8B2 B3                 OR   E
C8B3 2805               JR   Z,PA3
C8B5 25                 DEC  H
C8B6 20F8               JR   NZ,PA2
C8B8 18E0               JR   PASB
C8BA C9        PA3:     RET
               ;
C8BB E5        PLAY:    PUSH HL              ;HL=POINTER
C8BC D5        PY1:     PUSH DE              ;DATA OF LENGTH
C8BD CDC7C8             CALL PLSB            ;PLAY SUB
C8C0 D1                 POP  DE
C8C1 0D                 DEC  C
C8C2 20F8               JR   NZ,PY1
C8C4 E1                 POP  HL              ;HL=POINTER
C8C5 23                 INC  HL
C8C6 C9                 RET
               ;
C8C7 60        PLSB:    LD   H,B             ;H=SCALE CODE
C8C8 3A67EA             LD   A,(WOUT40)
C8CB CBEF               SET  5,A             ;BEEP 1
C8CD D340               OUT  (40H),A
C8CF 1B        PL1:     DEC  DE
C8D0 7A                 LD   A,D
C8D1 B3                 OR   E
C8D2 2815               JR   Z,PL3           ;LENGTH END
C8D4 25                 DEC  H
C8D5 20F8               JR   NZ,PL1
C8D7 60                 LD   H,B
C8D8 3A67EA             LD   A,(WOUT40)
C8DB CBAF               RES  5,A             ;BEEP 0
C8DD D340               OUT  (40H),A
C8DF 1B        PL2:     DEC  DE
C8E0 7A                 LD   A,D
C8E1 B3                 OR   E
C8E2 2805               JR   Z,PL3
C8E4 25                 DEC  H
C8E5 20F8               JR   NZ,PL2
C8E7 18DE               JR   PLSB
C8E9 C9        PL3:     RET
               ;
C8EA C5        TIME:    PUSH BC
C8EB 0E00               LD   C,0
C8ED 0D        TI1:     DEC  C
C8EE 20FD               JR   NZ,TI1
C8F0 C1                 POP  BC
C8F1 C9                 RET
               ;
                        END
```

第228図　UFOのチャルメラ屋さん（マシン語サブルーチン）

　　　ＵＳＲ２…チャルメラを鳴らす

これを追加します。そしてＵＦＯが向きを変えるところ，

　　1170行：Ｕ＝ＵＳＲ２（0）

　　1210行：Ｕ＝ＵＳＲ２（0）

でチャルメラを鳴らしてやれば，メイン・ルーチンは完成です（**第229図**）。

　それでは，さっそく

　　　ＵＦＯのチャルメラ屋さん

開始です。第228図，第229図の両方を入力し，

　　　　ＲＵＮ↘

でスタートです。まず，とぼけたＵＦＯが右に向かいます（**写真19**）。やがて右端に達したＵＦＯは，チャルメラを奏でます（**写真20**）。そして，また左に向かい（**写真21**），左端に達するとまたチャルメラです（**写真22**）。

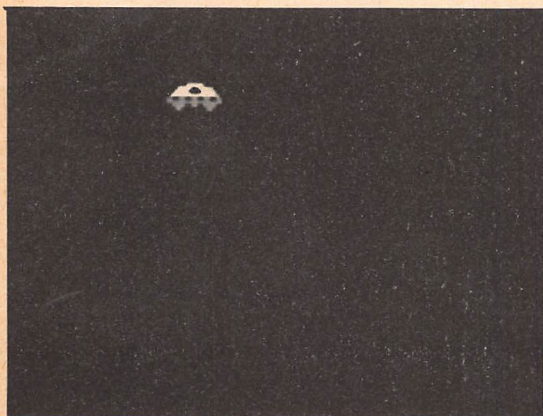
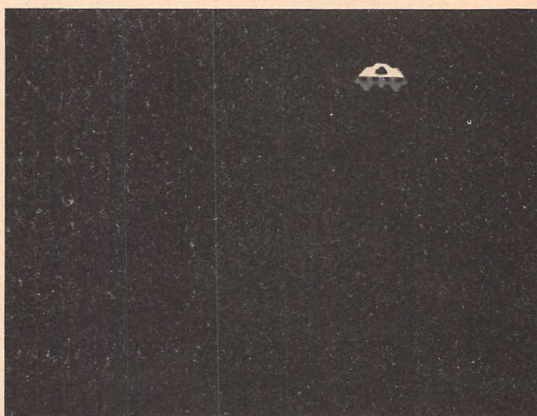
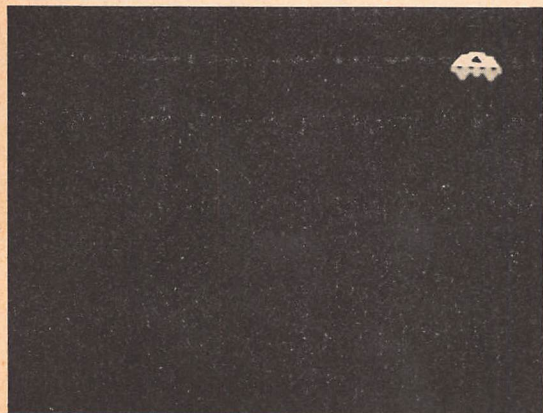
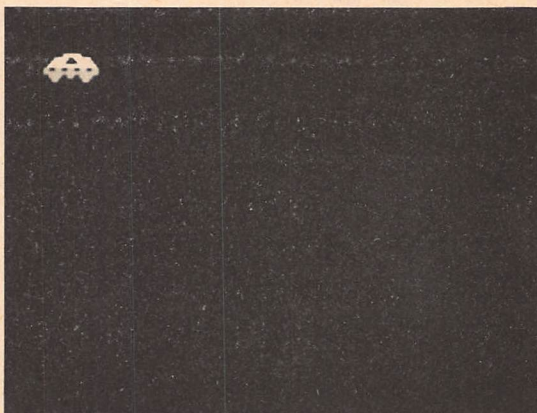
　ハハハ，何と愉快なことではありませんか。

　バンザーイ！

```
1000 '================================
1010 '  MOVING UFO-6
1020 '     1982.12.4:BY K.TSUKAGOSHI
1030 '================================
1040 '
1050 CLEAR 300,&HC7FF:DEFINT A-Z'                          ;SET USR FUNCTION
1060 DEF USR0=&HC800:DEF USR1=&HC81E:DEF USR2=&HC84C
1070 WIDTH 80,25:CONSOLE 0,25,0,1'                         ;SET TV MODE
1080 COLOR 7,0,1:PRINT CHR$(12)'                           ;SET COLOR GRAPHIC
1090 '
1100 LINE (0, 8)-(157,11),PRESET,4,BF'                     ;GREEN
1110 LINE (0,12)-(157,15),PRESET,1,BF'                     ;BLUE
1120 '
1130 X=0:Y=2:U=USR0(X*256+Y)'                              ;SET UFO AT LOCATE 1,2
1140 FOR X=0 TO 69'                                        ;MOVE UFO RIGHT
1150   U=USR1(X*256+Y):U=USR0((X+1)*256+Y)
1160   FOR J=0 TO 50:NEXT'                                 ;TIMER
1170 NEXT:U=USR2(0)
1180 FOR X=70 TO 1 STEP -1'                                ;MOVE UFO LEFT
1190   U=USR1(X*256+Y):U=USR0((X-1)*256+Y)
1200   FOR J=0 TO 50:NEXT'                                 ;TIMER
1210 NEXT:U=USR2(0)
1220 GOTO 1130'                                            ;LOOP FOREVER
```

チャルメラ演奏ルーチン

チャルメラを奏でる

第229図　UFOのチャルメラ屋さん（メイン）

《写真19》UFO右に向かう

《写真21》UFO左に向かう

《写真20》右端でチャルメラ演奏

《写真22》左端でチャルメラ演奏

実行速度を上げる

以上が，音楽演奏の基本部分です。好きなデータ列を用意すれば，好みの曲を演奏させることができます。また，サウンド・ルーチンの改良や音階データの改訂で，もっともっと面白い音を作れるでしょう．ゲームの効果音等に御利用ください。

さて，いよいよ**大詰**になりました。最後に**第230図**のプログラムを入力し，

GC100↘

で走らせてください。お馴染み〝選手を迎える歌〟です。ゲームで勝った時の効果音として有名ですね。

ところで，この音色，先に我々の作った〝チャルメラ〟と比べてかなり美しいことに気がつかれたと思います。

しかし第230図のサウンド・ルーチンは，我々が先に作ったルーチンとまったく同じものです。どうして音色にこのような差が出るのでしょう。

```
                    ;==============================
                    ;  MUSIC (SENSYU)
                    ;     82.12.5:BY K.TSUKAGOSHI
                    ;==============================
                    ;
                            ORG   0C100H
                    ;
093A                WID:    EQU   93AH
5C66                MON:    EQU   5C66H
EA67                WOUT40:EQU    0EA67H
                    ;
C100 AF                     XOR   A                ;STOP DMA      ] DMAをSTOPさせる
C101 D351                   OUT   (51H),A
C103 11FF30                 LD    DE,30FFH
C106 2115C1                 LD    HL,SENSYU
C109 CD58C1                 CALL  SONG
C10C 011950                 LD    BC,5019H         ;WIDTH 80,25   ] DMAをSTARTさせる
C10F CD3A09                 CALL  WID
C112 C3665C                 JP    MON
                    ;
C115 22042803 SENSYU:DB    22H,4,28H,3,26H,1,22H,4,33H,4
C119 26012204
C11D 3304
C11F 2D012801               DB    2DH,1,28H,1,26H,1,22H,1,26H,2,28H,2,2DH,1
C123 26012201
C127 26022802
C12B 2D01
C12D 44014401               DB    44H,1,44H,1,3DH,1,44H,1,4CH,1,51H,1,5BH,1
C131 3D014401
C135 4C015101
C139 5B01
C13B 28012601               DB    28H,1,26H,1,22H,1,1EH,1,22H,2,22H,2
C13F 22011E01
C143 22022202
C147 19042204               DB    19H,4,22H,4,26H,2,28H,1,2DH,1,2DH,3
C14B 26022801
C14F 2D012D03
C153 33013306               DB    33H,1,33H,6,0
C157 00
                    ;
C158 7E             SONG:   LD    A,(HL)           ;IN(HL=MUSIC DATA)
C159 A7                     AND   A
C15A C8                     RET   Z
C15B 47                     LD    B,A              ;B=SCALE CODE
C15C 23                     INC   HL
C15D 4E                     LD    C,(HL)           ;C=LENGTH
C15E 07                     RLCA                   ;bit7=1(PAUSE) ?
C15F 3005                   JR    NC,S01
C161 CD79C1                 CALL  PAUSE
C164 1803                   JR    S02
C166 CDA6C1         S01:    CALL  PLAY
C169 3A67EA         S02:    LD    A,(WOUT40)       ;BEEP 0
C16C CBAF                   RES   5,A
C16E D340                   OUT   (40H),A
C170 0610                   LD    B,10H
C172 CDD5C1         S03:    CALL  TIME
C175 10FB                   DJNZ  S03
C177 18DF                   JR    SONG
```

```
                   ;
C179 E5            PAUSE:  PUSH HL
C17A D5            PS1:    PUSH DE
C17B CD85C1                CALL PASB          ;PAUSE SUB
C17E D1                    POP  DE
C17F 0D                    DEC  C
C180 20F8                  JR   NZ,PS1
C182 E1                    POP  HL
C183 23                    INC  HL
C184 C9                    RET
                   ;
C185 60            PASB:   LD   H,B
C186 3A67EA                LD   A,(WOUT40)
C189 CBAF                  RES  5,A
C18B D340                  OUT  (40H),A
C18D 1B            PA1:    DEC  DE
C18E 7A                    LD   A,D
C18F B3                    OR   E
C190 2813                  JR   Z,PA3
C192 25                    DEC  H
C193 20F0                  JR   NZ,PASB
C195 60                    LD   H,B
C196 3A67EA                LD   A,(WOUT40)
C199 CBAF                  RES  5,A
C19B 1B            PA2:    DEC  DE
C19C 7A                    LD   A,D
C19D B3                    OR   E
C19E 2805                  JR   Z,PA3
C1A0 25                    DEC  H
C1A1 20F8                  JR   NZ,PA2
C1A3 18E0                  JR   PASB
C1A5 C9            PA3:    RET
                   ;
C1A6 E5            PLAY:   PUSH HL            ;HL=POINTER
C1A7 D5            PY1:    PUSH DE            ;DATA OF LENGTH
C1A8 CDB2C1                CALL PLSB          ;PLAY SUB
C1AB D1                    POP  DE
C1AC 0D                    DEC  C
C1AD 20F8                  JR   NZ,PY1
C1AF E1                    POP  HL            ;HL=POINTER
C1B0 23                    INC  HL
C1B1 C9                    RET
                   ;
C1B2 60            PLSB:   LD   H,B           ;H=SCALE CODE
C1B3 3A67EA                LD   A,(WOUT40)
C1B6 CBEF                  SET  5,A           ;BEEP 1
C1B8 D340                  OUT  (40H),A
C1BA 1B            PL1:    DEC  DE
C1BB 7A                    LD   A,D
C1BC B3                    OR   E
C1BD 2815                  JR   Z,PL3         ;LENGTH END
C1BF 25                    DEC  H
C1C0 20F8                  JR   NZ,PL1
C1C2 60                    LD   H,B
C1C3 3A67EA                LD   A,(WOUT40)
C1C6 CBAF                  RES  5,A           ;BEEP 0
C1C8 D340                  OUT  (40H),A
C1CA 1B            PL2:    DEC  DE
C1CB 7A                    LD   A,D
C1CC B3                    OR   E
C1CD 2805                  JR   Z,PL3
C1CF 25                    DEC  H
C1D0 20F8                  JR   NZ,PL2
C1D2 18DE                  JR   PLSB
C1D4 C9            PL3:    RET
                   ;
C1D5 C5            TIME:   PUSH BC
C1D6 0E00                  LD   C,0
C1D8 0D            TI1:    DEC  C
C1D9 20FD                  JR   NZ,TI1
C1DB C1                    POP  BC
C1DC C9                    RET
                   ;
                           END
```

第230図　DMAをOFFにして

これは，**DMAをSTOPさせる**という有名な方法を使っているからです。最後に，この方法を御紹介して第二巻はお別れすることに致しましょう。

DMAとは，

ダイレクト・メモリ・アクセス

(Direct Memory Access)

の略です。メモリからメモリ，メモリとI／O間の間で直接データのやり取りをさせる方法で，PC-8001では**ビデオRAMのデータをTV画面に転送する**のに使われています。DMAが行われている間は，**CPUがHOLD（STOP）状態**になっていますから，プログラムの実行は中断しています。

前に音楽演奏をさせるなら，実行速度は速い程良いと書きました。そこで，

DMAをSTOPさせる

とどういうことになるでしょうか？　**CPUにSTOPがかかりません**から，その分処理速度が向上します。ただし，ビデオRAMのデータ転送がSTOPしてしまいますから，

TV画面から映像が消去

してしまいますが，――。

効果音付きのカラー・グラフィック

DMAを止めるには，I／Oアドレスの**51H**にデータ**00H**を出力してやればできます。

```
XOR  A
OUT  (51H), A
```

ですね。

DMAを復活させるのは，かなり複雑です。長いルーチンを組まなければなりません。しかし，簡便法があります。

WIDTH

を使うと，自動的にDMAが再設定されますから，このことを利用します。BASICなら

WIDTH　80

等でDMAを復活できます。マシン語で行うなら，

Bレジスタ←画面ヨコ幅

Cレジスタ←画面タテ幅

をセットし，

CALL　93AH

でできます。第230図では，

```
LD    BC, 5019H
CALL  WID
```

で80×25サイズに設定し，DMAを復活させています。

いかがでしたか？　GAME等の効果音では，**美しい音の効果音**がほしい時があります。そんな時は，

DAMをSTOP

してやれば良いのです。ただし，画面が消えてしまいますから，GAMEの構成を良く考え，タイミング良くこの手法を使ってみてください。あなたのソフトが，以前にも増して光輝くものとなることでしょう。そう，いまやあなたは

効果音付きのカラー・グラフィック

があやつれるようになったのです。

あとがき

マシン語の基礎から，マシン語版スペース・インベーダーの製作までを目指す本シリーズの二冊目，いかがでしたか？

今回は，いちおう

電子音楽とカラー・グラフィック

を使えるようになるまでを狙ってみました。ここまでの知識だけでも，一応マシン語のゲームは作れます。ぜひ実験，実験を繰り返し，自分のプログラミングを発見し，ソフトの製作を進めていかれるようお勧め致します。

さて，次回**第三冊目の予告**です。まだ我々の知識だけでは，マシン語版インベーダーをこなすには，**ギャップ**があるようです。Z－80の各命令毎の研究も必要でしょう。各周辺機器の制御法も必要でしょう。またＰＣ－8800をお持ちの方へのサポートも必要かもしれません。それに，――も，――も，――。

これらすべてのギャップを埋めてからインベーダーに取りかかっていると，いつになったら最終目標に到達できるかわかりません。そこで，次の第三冊目は，ズバリ

マシン語版

スペース・インベーダー

の解析

です。いわば，本シリーズ最終目標の先取りというわけです。そして，第二巻と第三巻のギャップをうめる仕事はその後のことということになります。もっとも，それを必要とするか否かは，あなたが決定することですが――。

ま，いずれにしても本書一冊の読破，本当に御苦労様でした。読み始められたすべての人がここまでたどりつけたことを，そして本書に誤りが無いことを祈りつつ，まだまだ私からのささやかなプレゼントです。

〈本日の日付〉

年　　月　　日読破

サイン

塚越一雄

付録

《付録1-ⓐ》Z-80活用表

8ビット

×	A	B	C	D	E	H	L	(HL)	(BC)	(DE)	(IX +d)	(IY +d)	n	(nn)	I	R
LD A, ×	7F	78	79	7A	7B	7C	7D	7E	0A	1A	DD 7E d	FD 7E d	3E n	3A n n	ED 57	ED 5F
LD B, ×	47	40	41	42	43	44	45	46			DD 46 d	FD 46 d	06 n			
LD C, ×	4F	48	49	4A	4B	4C	4D	4E			DD 4E d	FD 4E d	0E n			
LD D, ×	57	50	51	52	53	54	55	56			DD 56 d	FD 56 d	16 n			
LD E, ×	5F	58	59	5A	5B	5C	5D	5E			DD 5E d	FD 5E d	1E n			
LD H, ×	67	60	61	62	63	64	65	66			DD 66 d	FD 66 d	26 n			
LD L, ×	6F	68	69	6A	6B	6C	6D	6E			DD 6E d	FD 6E d	2E n			
LD (HL), ×	77	70	71	72	73	74	75						36 n			
LD (BC), ×	02															
LD (DE), ×	12															
LD (IX+d), ×	DD 77 d	DD 70 d	DD 71 d	DD 72 d	DD 73 d	DD 74 d	DD 75 d						DD 36 d n			
LD (IY+d), ×	FD 77 d	FD 70 d	FD 71 d	FD 72 d	FD 73 d	FD 74 d	FD 75 d						FD 36 d n			
LD (nn), ×	32 n n															
LD I, ×	ED 47															
LD R, ×	ED 4F															
ADD A, ×	87	80	81	82	83	84	85	86			DD 86 d	FD 86 d	C6 n			
ADC A, ×	8F	88	89	8A	8B	8C	8D	8E			DD 8E d	FD 8E d	CE n			
SUB ×	97	90	91	92	93	94	95	96			DD 96 d	FD 96 d	D6 n			
SBC A, ×	9F	98	99	9A	9B	9C	9D	9E			DD 9E d	FD 9E d	DE n			
AND ×	A7	A0	A1	A2	A3	A4	A5	A6			DD A6 d	FD A6 d	E6 n			
XOR ×	AF	A8	A9	AA	AB	AC	AD	AE			DD AE d	FD AE d	EE n			
OR ×	B7	B0	B1	B2	B3	B4	B5	B6			DD B6 d	FD B6 d	F6 n			
CP ×	BF	B8	B9	BA	BB	BC	BD	BE			DD BE d	FD BE d	FE n			
INC ×	3C	04	0C	14	1C	24	2C	34			DD 34 d	FD 34 d				
DEC ×	3D	05	0D	15	1D	25	2D	35			DD 35 d	FD 35 d				

《付録1-ⓑ》Z-80活用表

回　転

＼×	A	B	C	D	E	H	L	(HL)	(IX+d)	(IY+d)
RLC ×	CB 07	CB 00	CB 01	CB 02	CB 03	CB 04	CB 05	CB 06	DD CB d 06	FD CB d 06
RRC ×	CB 0F	CB 08	CB 09	CB 0A	CB 0B	CB 0C	CB 0D	CB 0E	DD CB d 0E	FD CB d 0E
RL ×	CB 17	CB 10	CB 11	CB 12	CB 13	CB 14	CB 15	CB 16	DD CB d 16	FD CB d 16
RR ×	CB 1F	CB 18	CB 19	CB 1A	CB 1B	CB 1C	CB 1D	CB 1E	DD CB d 1E	FD CB d 1E
SLA ×	CB 27	CB 20	CB 21	CB 22	CB 23	CB 24	CB 25	CB 26	DD CB d 26	FD CB d 26
SRA ×	CB 2F	CB 28	CB 29	CB 2A	CB 2B	CB 2C	CB 2D	CB 2E	DD CB d 2E	FD CB d 2E
SRL ×	CB 3F	CB 38	CB 39	CB 3A	CB 3B	CB 3C	CB 3D	CB 3E	DD CB d 3E	FD CB d 3E
RLD								ED 6F		
RRD								ED 67		

RLCA	07
RRCA	0F
RLA	17
RRA	1F

16ビット

＼×	BC	DE	HL	SP	IX	IY	AF	nn	(nn)
LD AF, ×									
LD BC, ×								01 n n	ED 4B n n
LD DE, ×								11 n n	ED 5B n n
LD HL, ×								21 n n	2A n n
LD SP, ×			F9		DD F9	FD F9		31 n n	ED 7B n n
LD IX, ×								DD 21 n n	DD 2A n n
LD IY, ×								FD 21 n n	FD 2A n n
LD (nn), ×	ED 43 n n	ED 53 n n	22 n n	ED 73 n n	DD 22 n n	FD 22 n n			
PUSH ×	C5	D5	E5		DD E5	FD E5	F5		
POP ×	C1	D1	E1		DD E1	FD E1	F1		
ADD HL, ×	09	19	29	39					
ADD IX, ×	DD 09	DD 19		DD 39	DD 29				
ADD IY, ×	FD 09	FD 19		FD 39		FD 29			
ADC HL, ×	ED 4A	ED 5A	ED 6A	ED 7A					
SBC HL, ×	ED 42	ED 52	ED 62	ED 72					
INC ×	03	13	23	33	DD 23	FD 23			
DEC ×	0B	1B	2B	3B	DD 2B	FD 2B			

《付録1-©》Z-80活用表

ジャンプ，コール，リターン

×	UN COND	C	NC	Z	NZ	PE	PO	M	P	
JP ×, nn	C3 n n	DA n n	D2 n n	CA n n	C2 n n	EA n n	E2 n n	FA n n	F2 n n	
JR ×, e	18 e−2	38 e−2	30 e−2	28 e−2	20 e−2					
JP (HL)	E9									
JP (IX)	DD E9									
JP (IY)	FD E9									
CALL ×, nn	CD n n	DC n n	D4 n n	CC n n	C4 n n	EC n n	E4 n n	FC n n	F4 n n	
DJNZ e										10 e−2
RET ×	C9	D8	D0	C8	C0	E8	E0	F8	F0	
RETI	ED 4D									
RETN	ED 45									

入　力

IN A, n	DB n
IN A, (C)	ED 78
IN B, (C)	ED 40
IN C, (C)	ED 48
IN D, (C)	ED 50
IN E, (C)	ED 58
IN H, (C)	ED 60
IN L, (C)	ED 68
INI	ED A2
INIR	ED B2
IND	ED AA
INDR	ED BA

ブロック・サーチ

CPI	ED A1
CPIR	ED B1
CPD	ED A9
CPDR	ED B9

ブロック転送

LDI	ED A0
LDIR	ED B0
LDD	ED A8
LDDR	ED B8

アキュムレータ操作

DAA	27
CPL	2F
NEG	ED 44
CCF	3F
SCF	37

エクスチェンジ

EX AF, AF'	08
EX DE, HL	EB
EX (SP), HL	E3
EX (SP), IX	DD E3
EX (SP), IY	FD E3
EXX	D9

リスタート

RST 00H	C7
RST 08H	CF
RST 10H	D7
RST 18H	DF
RST 20H	E7
RST 28H	EF
RST 30H	F7
RST 38H	FF

CPUコントロール

NOP	00
HALT	76
DI	F3
EI	FB
IM 0	ED 46
IM 1	ED 56
IM 2	ED 5E

出　力

OUT n, A	D3 n
OUT (C), A	ED 79
OUT (C), B	ED 41
OUT (C), C	ED 49
OUT (C), D	ED 51
OUT (C), E	ED 59
OUT (C), H	ED 61
OUT (C), L	ED 69
OUTI	ED A3
OTIR	ED B3
OUTD	ED AB
OTDR	ED BB

《付録1-ⓓ》Z-80活用法

ビット操作

×	A	B	C	D	E	H	L	(HL)	(I× +d)	(IY +d)
BIT 0, ×	CB 47	CB 40	CB 41	CB 42	CB 43	CB 44	CB 45	CB 46	DD CB d 46	FD CB d 46
BIT 1, ×	CB 4F	CB 48	CB 49	CB 4A	CB 4B	CB 4C	CB 4D	CB 4E	DD CB d 4E	FD CB d 4E
BIT 2, ×	CB 57	CB 50	CB 51	CB 52	CB 53	CB 54	CB 55	CB 56	DD CB d 56	FD CB d 56
BIT 3, ×	CB 5F	CB 58	CB 59	CB 5A	CB 5B	CB 5C	CB 5D	CB 5E	DD CB d 5E	FD CB d 5E
BIT 4, ×	CB 67	CB 60	CB 61	CB 62	CB 63	CB 64	CB 65	CB 66	DD CB d 66	FD CB d 66
BIT 5, ×	CB 6F	CB 68	CB 69	CB 6A	CB 6B	CB 6C	CB 6D	CB 6E	DD CB d 6E	FD CB d 6E
BIT 6, ×	CB 77	CB 70	CB 71	CB 72	CB 73	CB 74	CB 75	CB 76	DD CB d 76	FD CB d 76
BIT 7, ×	CB 7F	CB 78	CB 79	CB 7A	CB 7B	CB 7C	CB 7D	CB 7E	DD CB d 7E	FD CB d 7E
RES 0, ×	CB 87	CB 80	CB 81	CB 82	CB 83	CB 84	CB 85	CB 86	DD CB d 86	FD CB d 86
RES 1, ×	CB 8F	CB 88	CB 89	CB 8A	CB 8B	CB 8C	CB 8D	CB 8E	DD CB d 8E	FD CB d 8E
RES 2, ×	CB 97	CB 90	CB 91	CB 92	CB 93	CB 94	CB 95	CB 96	DD CB d 96	FD CB d 96
RES 3, ×	CB 9F	CB 98	CB 99	CB 9A	CB 9B	CB 9C	CB 9D	CB 9E	DD CB d 9E	FD CB d 9E
RES 4, ×	CB A7	CB A0	CB A1	CB A2	CB A3	CB A4	CB A5	CB A6	DD CB d A6	FD CB d A6
RES 5, ×	CB AF	CB A8	CB A9	CB AA	CB AB	CB AC	CB AD	CB AE	DD CB d AE	FD CB d AE
RES 6, ×	CB B7	CB B0	CB B1	CB B2	CB B3	CB B4	CB B5	CB B6	DD CB d B6	FD CB d B6
RES 7, ×	CB BF	CB B8	CB B9	CB BA	CB BB	CB BC	CB BD	CB BE	DD CB d BE	FD CB d BE
SET 0, ×	CB C7	CB C0	CB C1	CB C2	CB C3	CB C4	CB C5	CB C6	DD CB d C6	FD CB d C6
SET 1, ×	CB CF	CB C8	CB C9	CB CA	CB CB	CB CC	CB CD	CB CE	DD CB d CE	FD CB d CE
SET 2, ×	CB D7	CB D0	CB D1	CB D2	CB D3	CB D4	CB D5	CB D6	DD CB d D6	FD CB d D6
SET 3, ×	CB DF	CB D8	CB D9	CB DA	CB DB	CB DC	CB DD	CB DE	DD CB d DE	FD CB d DE
SET 4, ×	CB E7	CB E0	CB E1	CB E2	CB E3	CB E4	CB E5	CB E6	DD CB d E6	FD CB d E6
SET 5, ×	CB EF	CB E8	CB E9	CB EA	CB EB	CB EC	CB ED	CB EE	DD CB d EE	FD CB d EE
SET 6, ×	CB F7	CB F0	CB F1	CB F2	CB F3	CB F4	CB F5	CB F6	DD CB d F6	FD CB d F6
SET 7, ×	CB FF	CB F8	CB F9	CB FA	CB FB	CB FC	CB FD	CB FE	DD CB d FE	FD CB d FE

《付録2-ⓐ》機械語↔ニーモニック対応表

機械語 ——→ ニーモニック			
00 NOP	40 LD B,B	80 ADD A, B	C0 RET NZ
01 LD BC,nn	41 LD B,C	81 ADD A, C	C1 POP BC
02 LD (BC),A	42 LD B,D	82 ADD A, D	C2 JP NZ,nn
03 INC BC	43 LD B,E	83 ADD A, E	C3 JP nn
04 INC B	44 LD B,H	84 ADD A, H	C4 CALL NZ,nn
05 DEC B	45 LD B,L	85 ADD A, L	C5 PUSH BC
06 LD B,n	46 LD B,(HL)	86 ADD A,(HL)	C6 ADD A,n
07 RLCA	47 LD B,A	87 ADD A, A	C7 RST 00H
08 EX AF,AF'	48 LD C,B	88 ADC A, B	C8 RET Z
09 ADD HL,BC	49 LD C,C	89 ADC A, C	C9 RET
0A LD A,(BC)	4A LD C'D	8A ADC A, D	CA JP Z,nn
0B DEC BC	4B LD C'E	8B ADC A, E	CB
0C INC C	4C LD C'H	8C ADC A, H	CC CALL Z,nn
0D DEC C	4D LD C'L	8D ADC A, L	CD CALL nn
0E LD C,n	4E LD C'(HL)	8E ADC A,(HL)	CE ADC A,n
0F RRCA	4F LD C'A	8F ADC A, A	CF RST 08H
10 DJNZ e	50 LD D,B	90 SUB B	D0 RET NC
11 LD DE,nn	51 LD D,C	91 SUB C	D1 POP DE
12 LD (DE),A	52 LD D,D	92 SUB D	D2 JP NC,nn
13 INC DE	53 LD D,E	93 SUB E	D3 OUT n,A
14 INC D	54 LD D,H	94 SUB H	D4 CALL NC,nn
15 DEC D	55 LD D,L	95 SUB L	D5 PUSH DE
16 LD D,n	56 LD D,(HL)	96 SUB (HL)	D6 SUB n
17 RLA	57 LD D,A	97 SUB A	D7 RST 10H
18 JR e	58 LD E,B	98 SBC A,B	D8 RET C
19 ADD HL,DE	59 LD E,C	99 SBC A,C	D9 EXX
1A LD A,(DE)	5A LD E,D	9A SBC A,D	DA JP C,nn
1B DEC DE	5B LD E,E	9B SBC A,E	DB IN A,n
1C INC E	5C LD E,H	9C SBC A,H	DC CALL C,nn
1D DEC E	5D LD E,L	9D SBC A,L	DD
1E LD E,n	5E LD E,(HL)	9E SBC A,(HL)	DE SBC A,n
1F RRA	5F LD E,A	9F SBC A,A	DF RST 18H
20 JR NZ,e	60 LD H,B	A0 AND B	E0 RET PO
21 LD HL,nn	61 LD H,C	A1 AND C	E1 POP HL
22 LD (nn),HL	62 LD H,D	A2 AND D	E2 JP PO,nn
23 INC HL	63 LD H,E	A3 AND E	E3 EX (SP),HL
24 INC H	64 LD H,H	A4 AND H	E4 CALL PO,nn
25 DEC H	65 LD H,L	A5 AND L	E5 PUSH HL
26 LD H,n	66 LD H,(HL)	A6 AND (HL)	E6 AND n
27 DAA	67 LD H,A	A7 AND A	E7 RST 20H
28 JR Z,e	68 LD L,B	A8 XOR B	E8 RET PE
29 ADD HL,HL	69 LD L,C	A9 XOR C	E9 JP (HL)
2A LD HL,(nn)	6A LD L,D	AA XOR D	EA JP PE,nn
2B DEC HL	6B LD L,E	AB XOR E	EB EX DE,HL
2C INC L	6C LD L,H	AC XOR H	EC CALL PE,nn
2D DEC L	6D LD L,L	AD XOR L	ED
2E LD L,n	6E LD L,(HL)	AE XOR (HL)	EE XOR n
2F CPL	6F LD L,A	AF XOR A	EF RST 28H
30 JR NC,e	70 LD (HL),B	B0 OR B	F0 RET P
31 LD SP,nn	71 LD (HL),C	B1 OR C	F1 POP AF
32 LD (nn),A	72 LD (HL),D	B2 OR D	F2 JP P,nn
33 INC SP	73 LD (HL),E	B3 OR E	F3 DI
34 INC (HL)	74 LD (HL),H	B4 OR H	F4 CALL P,nn
35 DEC (HL)	75 LD (HL),L	B5 OR L	F5 PUSH AF
36 LD (HL),n	76 HALT	B6 OR (HL)	F6 OR n
37 SCF	77 LD (HL),A	B7 OR A	F7 RST 30H
38 JR C,e	78 LD A,B	B8 CP B	F8 RET M
39 ADD HL,SP	79 LD A,C	B9 CP C	F9 LD SP,HL
3A LD A,(nn)	7A LD A,D	BA CP D	FA JP M,nn
3B DEC SP	7B LD A,E	BB CP E	FB EI
3C INC A	7C LD A,H	BC CP H	FC CALL M,nn
3D DEC A	7D LD A,L	BD CP L	FD
3E LD A,n	7E LD A,(HL)	BE CP (HL)	FE CP n
3F CCF	7F LD A,A	BF CP A	FF RST 38H

《付録2-ⓑ》機械語↔ニーモニック対応表

CB xx			
00 RLC B	40 BIT 0,B	80 RES 0,B	C0 SET 0,B
01 RLC C	41 BIT 0,C	81 RES 0,C	C1 SET 0,C
02 RLC D	42 BIT 0,D	82 RES 0,D	C2 SET 0,D
03 RLC E	43 BIT 0,E	83 RES 0,E	C3 SET 0,E
04 RLC H	44 BIT 0,H	84 RES 0,H	C4 SET 0,H
05 RLC L	45 BIT 0,L	85 RES 0,L	C5 SET 0,L
06 RLC (HL)	46 BIT 0,(HL)	86 RES 0,(HL)	C6 SET 0,(HL)
07 RLC A	47 BIT 0,A	87 RES 0,A	C7 SET 0,A
08 RRC B	48 BIT 1.B	88 RES 1,B	C8 SET 1.B
09 RRC C	49 BIT 1,C	89 RES 1,C	C9 SET 1,C
0A RRC D	4A BIT 1,D	8A RES 1,D	CA SET 1,D
0B RRC E	4B BIT 1,E	8B RES 1,E	CB SET 1,E
0C RRC H	4C BIT 1,H	8C RES 1,H	CC SET 1,H
0D RRC L	4D BIT 1,L	8D RES 1,L	CD SET 1,L
0E RRC (HL)	4E BIT 1,(HL)	8E RES 1,(HL)	CE SET 1,(HL)
0F RRC A	4F BIT 1,A	8F RES 1,A	CF SET 1,A
10 RL B	50 BIT 2,B	90 RES 2,B	D0 SET 2,B
11 RL C	51 BIT 2,C	91 RES 2,C	D1 SET 2,C
12 RL D	52 BIT 2,D	92 RES 2,D	D2 SET 2,D
13 RL E	53 BIT 2,E	93 RES 2,E	D3 SET 2,E
14 RL H	54 BIT 2,H	94 RES 2,H	D4 SET 2,H
15 RL L	55 BIT 2,L	95 RES 2,L	D5 SET 2,L
16 RL (HL)	56 BIT 2,(HL)	96 RES 2,(HL)	D6 SET 2,(HL)
17 RL A	57 BIT 2,A	97 RES 2,A	D7 SET 2,A
18 RR B	58 BIT 3,B	98 RES 3,B	D8 SET 3,B
19 RR C	59 BIT 3,C	99 RES 3,C	D9 SET 3,C
1A RR D	5A BIT 3,D	9A RES 3,D	DA SET 3,D
1B RR E	5B BIT 3,E	9B RES 3,E	DB SET 3,E
1C RR H	5C BIT 3,H	9C RES 3,H	DC SET 3,H
1D RR L	5D BIT 3,L	9D RES 3,L	DD SET 3,L
1E RR (HL)	5E BIT 3,(HL)	9E RES 3,(HL)	DE SET 3,(HL)
1F RR A	5F BIT 3,A	9F RES 3,A	DF SET 3,A
20 SLA B	60 BIT 4,B	A0 RES 4,B	E0 SET 4,B
21 SLA C	61 BIT 4,C	A1 RES 4,C	E1 SET 4,C
22 SLA D	62 BIT 4,D	A2 RES 4,D	E2 SET 4,D
23 SLA E	63 BIT 4,E	A3 RES 4,E	E3 SET 4,E
24 SLA H	64 BIT 4,H	A4 RES 4.H	E4 SET 4,H
25 SLA L	65 BIT 4.L	A5 RES 4,L	E5 SET 4,L
26 SLA (HL)	66 BIT 4,(HL)	A6 RES 4,(HL)	E6 SET 4,(HL)
27 SLA A	67 BIT 4,A	A7 RES 4,A	E7 SET 4,A
28 SRA B	68 BIT 5,B	A8 RES 5,B	E8 SET 5,B
29 SRA C	69 BIT 5,C	A9 RES 5,C	E9 SET 5,C
2A SRA D	6A BIT 5,D	AA RES 5,D	EA SET 5,D
2B SRA E	6B BIT 5,E	AB RES 5,E	EB SET 5,E
2C SRA H	6C BIT 5,H	AC RES 5,H	EC SET 5,H
2D SRA L	6D BIT 5,L	AD RES 5,L	ED SET 5,L
2E SRA (HL)	6E BIT 5,(HL)	AE RES 5,(HL)	EE SET 5,(HL)
2F SRA A	6F BIT 5,A	AF RES 5,A	EF SET 5,A
30	70 BIT 6,B	B0 RES 6,B	F0 SET 6,B
31	71 BIT 6,C	B1 RES 6,C	F1 SET 6,C
32	72 BIT 6,D	B2 RES 6,D	F2 SET 6,D
33	73 BIT 6,E	B3 RES 6,E	F3 SET 6,E
34	74 BIT 6,H	B4 RES 6,H	F4 SET 6,H
35	75 BIT 6,L	B5 RES 6,L	F5 SET 6,L
36	76 BIT 6,(HL)	B6 RES 6,(HL)	F6 SET 6,(HL)
37	77 BIT 6,A	B7 RES 6,A	F7 SET 6,A
38 SRL B	78 BIT 7,B	B8 RES 7,B	F8 SET 7,B
39 SRL C	79 BIT 7,C	B9 RES 7,C	F9 SET 7,C
3A SRL D	7A BIT 7,D	BA RES 7,D	FA SET 7,D
3B SRL E	7B BIT 7,E	BB RES 7,E	FB SET 7,E
3C SRL H	7C BIT 7,H	BC RES 7,H	FC SET 7,H
3D SRL L	7D BIT 7,L	BD RES 7,L	FD SET 7,L
3E SRL (HL)	7E BIT 7,(HL)	BE RES 7,(HL)	FE SET 7,(HL)
3F SRL A	7F BIT 7,A	BF RES 7,A	FF SET 7,A

《付録2-©》機械語↔ニーモニック対応表

DD ××		
09	ADD	IX, BC
19	ADD	IX, DE
21	LD	IX, nn
22	LD	(nn), IX
23	INC	IX
29	ADD	IX, IX
2A	LD	IX, (nn)
2B	DEC	IX
34	INC	(IX+d)
35	DEC	(IX+d)
36	LD	(IX+d), n
39	ADD	IX, SP
46	LD	B, (IX+d)
4E	LD	C, (IX+d)
56	LD	D, (IX+d)
5E	LD	E, (IX+d)
66	LD	H, (IX+d)
6E	LD	L, (IX+d)
70	LD	(IX+d), B
71	LD	(IX+d), C
72	LD	(IX+d), D
73	LD	(IX+d), E
74	LD	(IX+d), H
75	LD	(IX+d), L
77	LD	(IX+d), A
7E	LD	A, (IX+d)
86	ADD	A, (IX+d)
8E	ADC	A, (IX+d)
96	SUB	(IX+d)
9E	SBC	A, (IX+d)
A6	AND	(IX+d)
AE	XOR	(IX+d)
B6	OR	(IX+d)
BE	CP	(IX+d)
CB d 06	RLC	(IX+d)
CB d 0E	RRC	(IX+d)
CB d 16	RL	(IX+d)
CB d 1E	RR	(IX+d)
CB d 26	SLA	(IX+d)
CB d 2E	SRA	(IX+d)
CB d 3E	SRL	(IX+d)
CB d 46	BIT	0, (IX+d)
CB d 4E	BIT	1, (IX+d)
CB d 56	BIT	2, (IX+d)
CB d 5E	BIT	3, (IX+d)
CB d 66	BIT	4, (IX+d)
CB d 6E	BIT	5, (IX+d)
CB d 76	BIT	6, (IX+d)
CB d 7E	BIT	7, (IX+d)
CB d 86	RES	0, (IX+d)
CB d 8E	RES	1, (IX+d)
CB d 96	RES	2, (IX+d)
CB d 9E	RES	3, (IX+d)
CB d A6	RES	4, (IX+d)
CB d AE	RES	5, (IX+d)
CB d B6	RES	6, (IX+d)
CB d BE	RES	7, (IX+d)
CB d C6	SET	0, (IX+d)
CB d CE	SET	1, (IX+d)
CB d D6	SET	2, (IX+d)
CB d DE	SET	3, (IX+d)
CB d E6	SET	4, (IX+d)
CB d EE	SET	5, (IX+d)
CB d F6	SET	6, (IX+d)
CB d FE	SET	7, (IX+d)
E1	POP	IX
E3	EX	(SP), IX
E5	PUSH	IX
E9	JP	(IX)
F9	LD	SP, IX

ED ××		
40	IN	B, (C)
41	OUT	(C), B
42	SBC	HL, BC
43	LD	(nn), BC
44	NEG	
45	RETN	
46	IM	0
47	LD	I, A
48	IN	C, (C)
49	OUT	(C), C
4A	ADC	HL, BC
4B	LD	BC, (nn)
4D	RETI	
4F	LD	R, A
50	IN	D, (C)
51	OUT	(C), D
52	SBC	HL, DE
53	LD	(nn), DE
56	IM	1
57	LD	A, I
58	IN	E, (C)
59	OUT	(C), E
5A	ADC	HL, DE
5B	LD	DE, (nn)
5E	IM	2
5F	LD	A, R
60	IN	H, (C)
61	OUT	(C), H
62	SBC	HL, HL
67	RRD	
68	IN	L, (C)
69	OUT	(C), L
6A	ADC	HL, HL
6F	RLD	
72	SBC	HL, SP
73	LD	(nn), SP
78	IN	A, (C)
79	OUT	(C), A
7A	ADC	HL, SP
7B	LD	SP, (nn)
A0	LDI	
A1	CPI	
A2	INI	
A3	OUTI	
A8	LDD	
A9	CPD	
AA	IND	
AB	OUTD	
B0	LDIR	
B1	CPIR	
B2	INIR	
B3	OTIR	
B8	LDDR	
B9	CPDR	
BA	INDR	
BB	OTDR	

FD ××		
09	ADD	IY, BC
19	ADD	IY, DE
21	LD	IY, nn
22	LD	(nn), IY
23	INC	IY
29	ADD	IY, IY
2A	LD	IY, (nn)
2B	DEC	IY
34	INC	(IY+d)
35	DEC	(IY+d)
36	LD	(IY+d), n
39	ADD	IY, SP
46	LD	B, (IY+d)
4E	LD	C, (IY+d)
56	LD	D, (IY+d)
5E	LD	E, (IY+d)
66	LD	H, (IY+d)
6E	LD	L, (IY+d)
70	LD	(IY+d), B
71	LD	(IY+d), C
72	LD	(IY+d), D
73	LD	(IY+d), E
74	LD	(IY+d), H
75	LD	(IY+d), L
77	LD	(IY+d), A
7E	LD	A, (IY+d)
86	ADD	A, (IY+d)
8E	ADC	A, (IY+d)
96	SUB	(IY+d)
9E	SBC	A, (IY+d)
A6	AND	(IY+d)
AE	XOR	(IY+d)
B6	OR	(IY+d)
BE	CP	(IY+d)
CB d 06	RLC	(IY+d)
CB d 0E	RRC	(IY+d)
CB d 16	RL	(IY+d)
CB d 1E	RR	(IY+d)
CB d 26	SLA	(IY+d)
CB d 2E	SRA	(IY+d)
CB d 3E	SRL	(IY+d)
CB d 46	BIT	0, (IY+d)
CB d 4E	BIT	1, (IY+d)
CB d 56	BIT	2, (IY+d)
CB d 5E	BIT	3, (IY+d)
CB d 66	BIT	4, (IY+d)
CB d 6E	BIT	5, (IY+d)
CB d 76	BIT	6, (IY+d)
CB d 7E	BIT	7, (IY+d)
CB d 86	RES	0, (IY+d)
CB d 8E	RES	1, (IY+d)
CB d 96	RES	2, (IY+d)
CB d 9E	RES	3, (IY+d)
CB d A6	RES	4, (IY+d)
CB d AE	RES	5, (IY+d)
CB d B6	RES	6, (IY+d)
CB d BE	RES	7, (IY+d)
CB d C6	SET	0, (IY+d)
CB d CE	SET	1, (IY+d)
CB d D6	SET	2, (IY+d)
CB d DE	SET	3, (IY+d)
CB d E6	SET	4, (IY+d)
CB d EE	SET	5, (IY+d)
CB d F6	SET	6, (IY+d)
CB d FE	SET	7, (IY+d)
E1	POP	IY
E3	EX	(SP), IY
E5	PUSH	IY
E9	JP	(IY)
F9	LD	SP, IY

《付録3》10進↔16進変換表

上位＼下位	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
2	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
3	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
4	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
5	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
6	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
7	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
8	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
9	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
A	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
B	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
C	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
D	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
E	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
F	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

桁＼最上位×	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
×00	256	512	768	1024	1280	1536	1792	2048	2304	2560	2816	3072	3328	3584	3840
×000	4096	8192	12288	16384	20480	24576	28672	32768	36864	40960	45056	49152	53248	57344	61440
×0000	65536														

《付録4》2 進↔16進変換表

16 進 数	2 進 数
0	0 0 0 0
1	0 0 0 1
2	0 0 1 0
3	0 0 1 1
4	0 1 0 0
5	0 1 0 1
6	0 1 1 0
7	0 1 1 1
8	1 0 0 0
9	1 0 0 1
A	1 0 1 0
B	1 0 1 1
C	1 1 0 0
D	1 1 0 1
E	1 1 1 0
F	1 1 1 1

《付録 5》命令のフラグへの影響

インストラクション	D7							D0	コメント
	S	Z		H		P/V	N	C	
ADD A,s;ADC A,s	↕	↕	×	↕	×	V	0	↕	8-bit add or add with carry
SUBs;SBC A,s;CPs;NEG	↕	↕	×	↕	×	V	1	↕	8-bit subtract, subtract with carry, compare and negate accumulator
ANDs	↕	↕	×	1	×	P	0	0	Logical operations
ORs;XORs	↕	↕	×	0	×	P	0	0	
INCs	↕	↕	×	↕	×	V	0	•	8-bit increment
DECs	↕	↕	×	↕	×	V	1	•	8-bit decrement
ADD DD,ss	•	•	×	×	×	•	0	↕	16-bit add
ADC HL,ss	↕	↕	×	×	×	V	0	↕	16-bit add with carry
SBC HL,ss	↕	↕	×	×	×	V	1	↕	16-bit subtract with carry
RLA;RLCA;RRA;RRCA	•	•	×	0	×	•	0	↕	Rotate accumulator
RLs;RLCs;RRs;RRCs; SLAs;SRAs;SRLs	↕	↕	×	0	×	P	0	↕	Rotate and shift locations
RLD;RRD	↕	↕	×	0	×	P	0	•	Rotate digit left and right
DAA	↕	↕	×	↕	×	P	•	↕	Decimal adjust accumulator
CPL	•	•	×	1	×	•	1	•	Complement accumulator
SCF	•	•	×	0	×	•	0	1	Set carry
CCF	•	•	×	×	×	•	0	↕	Complement carry
INr,(C)	↕	↕	×	0	×	P	0	•	Input register indirect
INI;IND;OUTI;OUTD	×	↕	×	×	×	×	1	•	Block input and output
INIR;INDR;OTIR;OTDR	×	1	×	×	×	×	1	•	Z=0 if B≠0 otherwise Z=1
LDI;LDD	×	×	×	0	×	↕	0	•	Block transfer instructions
LDIR;LDDR	×	×	×	0	×	0	0	•	P/V=1 if BC≠0, otherwise P/V=0
CPI;CPIR;CPD;CPDR	×	↕	×	×	×	↕	1	•	Block search instructions Z=1 if A=(HL), otherwise Z=0 P/V=1 if BC≠0, otherwise P/V=0
LD A,I;LD A,R	↕	↕	×	0	×	IFF	0	•	The content of the interrupt enable flip-flop(IFF) is copied into the P/V flag
BIT b,s	×	↕	×	1	×	×	0	•	The state of bit b of locations is copied into the Z flag

《付録6-ⓐ》Z-80 CODING SHEET

257H

ADRESS 1	MACHINE CODE 10	LEBEL 20	OPERATION & OPERAND 30 40	COMMENT 50 60 70 80
0275		OUT:	EQU 0275H	
D000	0650		LD B,50H	
D002	3E78		LD A,78H	
D004	CD7502		CALL OUT	
D006	1000D0		DJNZ D000H	
D008	C3665C		JP 5366H	
D00A	00		NOP	
D00B	00		NOP	
D00C	00		NOP	
D00D	00		NOP	
D00E	00		NOP	
D00F	00		NOP	

(ⓐ,ⓑは同じものです。B４で２頁まとめてコピーし，切断してご使用下さい）

《付録6-ⓑ》Z-80　CODING SHEET

(ⓐ, ⓑは同じものです。B 4 で 2 頁まとめてコピーし, 切断してご使用下さい)

《付録7》レイアウト・シート（40×25モード）

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
0	F3 00	02	04	06	08	0A	0C	0E	F3 10	12	14	16	18	1A	1C	1E	F3 20	22	24	26	28	2A	2C	2E	F3 30	32	34	36	38	3A	3C	3E	F3 40	42	44	46	48	4A	4C	4E
1	F3 78	7A	7C	7E	F3 80	82	84	86	88	8A	8C	8E	F3 90	92	94	96	98	9A	9C	9E	F3 A0	A2	A4	A6	A8	AA	AC	AE	F3 B0	B2	B4	B6	B8	BA	BC	BE	F3 C0	C2	C4	C6
2	F3 F0	F2	F4	F6	F8	FA	FC	FE	F4 00	02	04	06	08	0A	0C	0E	F4 10	12	14	16	18	1A	1C	1E	F4 20	22	24	26	28	2A	2C	2E	F4 30	32	34	36	38	3A	3C	3E
3	F4 68	6A	6C	6E	F4 70	72	74	76	78	7A	7C	7E	F4 80	82	84	86	88	8A	8C	8E	F4 90	92	94	96	98	9A	9C	9E	F4 A0	A2	A4	A6	A8	AA	AC	AE	F4 B0	B2	B4	B6
4	F4 E0	E2	E4	E6	E8	EA	EC	EE	F4 F0	F2	F4	F6	F8	FA	FC	FE	F5 00	02	04	06	28	0A	0C	0E	F5 10	12	14	16	18	1A	1C	1E	F5 20	22	24	26	28	2A	2C	2E
5	F5 58	5A	5C	5E	F5 60	62	64	66	68	6A	6C	6E	F5 70	72	74	76	78	7A	7C	7E	F5 08	82	84	86	88	8A	8C	8E	F5 90	92	94	96	98	9A	9C	9E	F5 A0	A2	A4	A6
6	F5 D0	D2	D4	D6	D8	DA	DC	DE	F5 E0	E2	E4	E6	E8	EA	EC	EE	F5 F0	F2	F4	F6	F8	FA	FC	FE	F6 00	02	04	06	08	0A	0C	0E	F6 10	12	14	16	18	1A	1C	1E
7	F6 48	4A	4C	4E	F6 50	52	54	56	58	5A	5C	5E	F6 60	62	64	66	68	6A	6C	6E	F6 70	72	74	76	78	7A	7C	7E	F6 80	82	84	86	88	8A	8C	8E	F6 90	92	94	96
8	F6 C0	C2	C4	C6	C8	CA	CC	CE	F6 D0	D2	D4	D6	D8	DA	DC	DE	F6 E0	E2	E4	E6	E8	EA	EC	EE	F6 F0	F2	F4	F6	F8	FA	FC	FE	F7 00	02	04	06	08	0A	0C	0E
9	F7 38	3A	3C	3E	F7 40	42	44	46	48	4A	4C	4E	F7 50	52	54	56	58	5A	5C	5E	F7 60	62	64	66	68	6A	6C	6E	F7 70	72	74	76	78	7A	7C	7E	F7 80	82	84	86
10	F7 B0	B2	B4	B6	B8	BA	BC	BE	F7 C0	C2	C4	C6	C8	CA	CC	CE	F7 D0	D2	D4	D6	D8	DA	DC	DE	F7 E0	E2	E4	E6	E8	EA	EC	EE	F7 F0	F2	F4	F6	F8	FA	FC	FE
11	F8 28	2A	2C	2E	F8 30	32	34	36	38	3A	3C	3E	F8 40	42	44	46	48	4A	4C	4E	F8 50	52	54	56	58	5A	5C	5E	F8 60	62	64	66	68	6A	6C	6E	F8 70	72	74	76
12	F8 A0	A2	A4	A6	A8	AA	AC	AE	F8 B0	B2	B4	B6	B8	BA	BC	BE	F8 C0	C2	C4	C6	C8	CA	CC	CE	F8 D0	D2	D4	D6	D8	DA	DC	DE	F8 E0	E2	E4	E6	E8	EA	EC	EE
13	F9 18	1A	1C	1E	F9 20	22	24	26	28	2A	2C	2E	F9 30	32	34	36	38	3A	3C	3E	F9 40	42	44	46	48	4A	4C	4E	F9 50	52	54	56	58	5A	5C	5E	F9 60	62	64	66
14	F9 90	92	94	96	98	9A	9C	9E	F9 A0	A2	A4	A6	A8	AA	AC	AE	F9 B0	B2	B4	B6	B8	BA	BC	BE	F9 C0	C2	C4	C6	C8	CA	CC	CE	F9 D0	D2	D4	D6	D8	DA	DC	DE
15	FA 08	0A	DC	0E	FA 10	12	14	16	18	1A	1C	1E	FA 20	22	24	26	28	2A	2C	2E	FA 30	32	34	36	38	3A	3C	3E	FA 40	42	44	46	48	4A	4C	4E	FA 50	52	54	56
16	FA 80	82	84	86	88	8A	8C	8E	FA 90	92	94	96	98	9A	9C	9E	FA A0	A2	A4	A6	A8	AA	AC	AE	FA B0	B2	B4	B6	B8	BA	BC	BE	FA C0	C2	C4	C6	C8	CA	CC	CE
17	FA F8	FA	FC	FE	FB 00	02	04	06	08	0A	0C	0E	FB 10	12	14	16	18	1A	1C	1E	FB 20	22	24	26	28	2A	2C	2E	FB 30	32	34	36	38	3A	3C	3E	FB 40	42	44	46
18	FB 70	72	74	76	78	7A	7C	7E	FB 80	82	84	86	88	8A	8C	8E	FB 90	92	94	96	98	9A	9C	9E	FB A0	A2	A4	A6	A8	AA	AC	AE	FB B0	B2	B4	B6	B8	BA	BC	BE
19	FB E8	EA	EC	EE	FB F0	F2	F4	F6	F8	FA	EC	FE	FC 00	02	04	06	08	DA	0C	0E	FC 10	12	14	16	18	1A	1C	1E	FC 20	22	24	26	28	2A	2C	2E	FC 30	32	34	36
20	FC 60	62	64	66	68	6A	6C	6E	FC 70	72	74	76	78	7A	7C	7E	FC 80	82	84	86	88	8A	8C	8E	FC 90	92	94	96	98	9A	9C	9E	FC A0	A2	A4	A6	A8	AA	AC	AE
21	FC D8	DA	DC	DE	FC E0	E2	E4	E6	E8	EA	EC	EE	FC F0	F2	F4	F6	F8	FA	FC	FE	FD 00	02	04	06	08	0A	0C	0E	FD 10	12	14	16	18	1A	1C	1E	FD 20	22	24	26
22	FD 50	52	54	56	58	5A	5C	5E	FD 60	62	64	66	68	6A	6C	6E	FD 70	72	74	76	78	7A	7C	7E	FD 80	82	84	86	88	8A	8C	8E	FD 90	92	94	96	98	9A	9C	9E
23	FD C8	CA	CC	CE	FD D0	D2	D4	D6	D8	CA	CC	CE	FD E0	E2	E4	E6	E8	EA	EC	EE	FD F0	F2	F4	F6	F8	FA	FC	FE	FE 00	02	04	06	08	0A	0C	0E	FE 10	12	14	16
24	FE 40	42	44	46	48	4A	4C	4E	FE 50	52	54	56	58	5A	5C	5E	FE 60	62	64	66	68	6A	6C	6E	FE 70	72	74	76	78	7A	7C	7E	FE 80	82	84	86	88	8A	8C	8E

《付録8》レイアウト・シート(80×25モード)

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	F3 00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	F3 10	11	12	13
1	F3 78	79	7A	7B	7C	7D	7E	7F	F3 80	81	82	83	84	85	86	87	88	89	8A	8B
2	F3 F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	F4 00	01	02	03
3	F4 68	69	6A	6B	6C	6D	6E	6F	F4 70	71	72	73	74	75	76	77	78	79	7A	7B
4	F4 E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F4 F0	F1	F2	F3
5	F5 58	59	5A	5B	5C	5D	5E	5F	F5 60	61	62	63	64	65	66	67	68	69	6A	6B
6	F5 D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	F5 E0	E1	E2	E3
7	F6 48	49	4A	4B	4C	4D	4E	4F	F6 50	51	52	53	54	55	56	57	58	59	5A	5B
8	F6 C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	F6 D0	D1	D2	D3
9	F7 38	39	3A	3B	3C	3D	3E	3F	F7 40	41	42	43	44	45	46	47	48	49	4A	4B
10	F7 B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	F7 C0	C1	C2	C3
11	F8 28	29	2A	2B	2C	2D	2E	2F	F8 30	31	32	33	34	35	36	37	38	39	3A	3B
12	F8 A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	F8 B0	B1	B2	B3
13	F9 18	19	1A	1B	1C	1D	1E	1F	F9 20	21	22	23	24	25	26	27	28	29	2A	2B
14	F9 90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	F9 A0	A1	A2	A3
15	FA 08	09	0A	0B	0C	0D	0E	0F	FA 10	11	12	13	14	15	16	17	18	19	1A	1B
16	FA 80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	FA 90	91	92	93
17	FA F8	F9	FA	FB	FC	FD	FE	FF	FB 00	01	02	03	04	05	06	07	08	09	0A	0B
18	FB 70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	FB 80	81	82	83
19	FB E8	E9	EA	EB	EC	ED	EE	EF	FB F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB
20	FC 60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	FC 70	71	72	73
21	FC D8	D9	DA	DB	DC	DD	DE	DF	FC E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB
22	FD 50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	FD 60	61	62	63
23	FD C8	C9	CA	CB	CC	CD	CE	CF	FD D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB
24	FE 40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	FE 50	51	52	53

	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39
0	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	F3 20	21	22	23	24	25	26	27
1	8C	8D	8E	8F	F3 90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
2	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	F4 10	11	12	13	14	15	16	17
3	7C	7D	7E	7F	F4 80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
4	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	F5 00	01	02	03	04	05	06	07
5	6C	6D	6E	6F	F5 70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
6	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF	F5 F0	F1	F2	F3	F4	F5	F6	F7
7	5C	5D	5E	5F	F6 60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
8	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	F6 E0	E1	E2	E3	E4	E5	E6	E7
9	4C	4D	4E	4F	F7 50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
10	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	F7 D0	D1	D2	D3	D4	D5	D6	D7
11	3C	3D	3E	3F	F8 40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
12	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	F8 C0	C1	C2	C3	C4	C5	C6	C7
13	2C	2D	2E	2F	F9 30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
14	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	F9 B0	B1	B2	B3	B4	B5	B6	B7
15	1C	1D	1E	1F	FA 20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
16	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	FA A0	A1	A2	A3	A4	A5	A6	A7
17	0C	0D	0E	0F	FB 10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
18	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	FB 90	91	92	93	94	95	96	97
19	FC	FD	FE	FF	FC 00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
20	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	FC 80	81	82	83	84	85	86	87
21	EC	ED	EE	EF	FC F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF
22	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	FD 70	71	72	73	74	75	76	77
23	DC	DD	DE	DF	FD E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
24	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	FE 60	61	62	63	64	65	66	67

	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
0	28	29	2A	2B	2C	2D	2E	2F	F3 30	31	32	33	34	35	36	37	38	39	3A	3B
1	F3 A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	F3 B0	B1	B2	B3
2	18	19	1A	1B	1C	1D	1E	1F	F4 20	21	22	23	24	25	26	27	28	29	2A	2B
3	F4 90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	F4 A0	A1	A2	A3
4	08	09	0A	0B	0C	0D	0E	0F	F5 10	11	12	13	14	15	16	17	18	19	1A	1B
5	F5 80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	F5 90	91	92	93
6	F8	F9	FA	FB	FC	FD	FE	FF	F6 00	01	02	03	04	05	06	07	08	09	0A	0B
7	F6 70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	F6 80	81	82	83
8	E8	E9	EA	EB	EC	ED	EE	EF	F6 F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB
9	F7 60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	F7 70	71	72	73
10	D8	D9	DA	DB	DC	DD	DE	DF	F7 E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB
11	F8 50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	F8 60	61	62	63
12	C8	C9	CA	CB	CC	CD	CE	CF	F8 D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB
13	F9 40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	F9 50	51	52	53
14	B8	B9	BA	BB	BC	BD	BE	BF	F9 C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB
15	FA 30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	FA 40	41	42	43
16	A8	A9	AA	AB	AC	AD	AE	AF	FA B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB
17	FB 20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	FB 30	31	32	33
18	98	99	9A	9B	9C	9D	9E	9F	FB A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB
19	FC 10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	FC 20	21	22	23
20	88	89	8A	8B	8C	8D	8E	8F	FC 90	91	92	93	94	95	96	97	98	99	9A	9B
21	FD 00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	FD 10	11	12	13
22	78	79	7A	7B	7C	7D	7E	7F	ED 80	81	82	83	84	85	86	87	88	89	8A	8B
23	FD F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF	FE 00	01	02	03
24	68	69	6A	6B	6C	6D	6E	6F	FE 70	71	72	73	74	75	76	77	78	79	7A	7B

	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
0	3C	3D	3E	3F	F3 40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
1	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF	F3 C0	C1	C2	C3	C4	C5	C6	C7
2	2C	2D	2E	2F	F4 30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF	F4 B0	B1	B2	B3	B4	B5	B6	B7
4	1C	1D	1E	1F	F5 20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
5	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	F5 A0	A1	A2	A3	A4	A5	A6	A7
6	0C	0D	0E	0F	F6 10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
7	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	F6 90	91	92	93	94	95	96	97
8	FC	FD	FE	FF	F7 00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
9	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F	F7 80	81	82	83	84	85	86	87
10	EC	ED	EE	EF	F7 F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF
11	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	F8 70	71	72	73	74	75	76	77
12	DC	DD	DE	DF	F8 E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
13	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	F9 60	61	62	63	64	65	66	67
14	CC	CD	CE	CF	F9 D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
15	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F	FA 50	51	52	53	54	55	56	57
16	BC	BD	BE	BF	FA C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
17	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	FB 40	41	42	43	44	45	46	47
18	AC	AD	AE	AF	FB B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
19	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	FC 30	31	32	33	34	35	36	37
20	9C	9D	9E	9F	FC A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
21	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F	FD 20	21	22	23	24	25	26	27
22	8C	8D	8E	8F	FD 90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
23	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	FE 10	11	12	13	14	15	16	17
24	7C	7D	7E	7F	FE 80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F

《付録9》μPD780インストラクション一覧表

μPD780のインストラクション・セットの概要一覧表を示します。ここでは，プログラミングで必要な情報と命令実行時間等を示してあります。次の欄には次のようなものが設けてあります。

○ニーモニック欄

○オペレーション欄

○フラグ欄：命令実行後のフラグ・レジスタの内容

○ＯＰコード欄：オペレーション・コード

○バイト欄

○マシン・サイクル欄

○ステート欄：命令のフェッチと実行に必要なステート数

○コメント欄　　　　　　　　　　　（＝外部クロック・サイクル）

μPD780のインストラクション・セットの概要一覧表を示します．ここでは，プログラミングで必要な情報と命令実行時間等を示してあります．表の欄には次のようなものが設けてあります．

○ニーモニック欄

○オペレーション欄

○フラグ欄　　　　：命令実行後のフラグ・レジスタの内容

○ＯＰコード欄　　：オペレーション・コード

○バイト欄

○マシン・サイクル欄

○ステート欄　　　：命令のフェッチと実行に必要なステート数

○コメント欄　　　　　　　　　　　（＝外部クロック・サイクル）

注：付録9のμＰＤ780インストラクション一覧表は日本電気(株)の「μCOM-82ユーザーズ・マニュアル」より転載

8ビット・ロード命令

ニーモニック	オペレーション	フラグ								OPコード		バイト	マシン・サイクル	ステート	コメント
		S	Z		H		P/V	N	C	76 543 210	Hex				
LD r,s	r←s	•	•	×	•	×	•	•	•	01 r s		1	1	4	r,s Reg
LD r,n	r←n	•	•	×	•	×	•	•	•	00 r 110		2	2	7	000 B
										← n →					001 C
LD r,(HL)	r←(HL)	•	•	×	•	×	•	•	•	01 r 110		1	2	7	010 D
LD r,(IX+d)	r←(IX+d)	•	•	×	•	×	•	•	•	11 011 101	DD	3	5	19	011 E
										01 r 110					100 H
										← d →					101 L
LD r,(IY+d)	r←(IY+d)	•	•	×	•	×	•	•	•	11 111 101	FD	3	5	19	111 A
										01 r 110					
										← d →					
LD (HL),r	(HL)←r	•	•	×	•	×	•	•	•	01 110 r		1	2	7	
LD (IX+d),r	(IX+d)←r	•	•	×	•	×	•	•	•	11 011 101	DD	3	5	19	
										01 110 r					
										← d →					
LD (IY+d),r	(IY+d)←r	•	•	×	•	×	•	•	•	11 111 101	FD	3	5	19	
										01 110 r					
										← d →					
LD (HL),n	(HL)←n	•	•	×	•	×	•	•	•	00 110 110	36	2	3	10	
										← n →					
LD (IX+d),n	(IX+d)←n	•	•	×	•	×	•	•	•	11 011 101	DD	4	5	19	
										00 110 110	36				
										← d →					
										← n →					
LD (IY+d),n	(IY+d)←n	•	•	×	•	×	•	•	•	11 111 101	FD	4	5	19	
										00 110 110	36				
										← d →					
										← n →					
LD A,(BC)	A←(BC)	•	•	×	•	×	•	•	•	00 001 010	0A	1	2	7	
LD A,(DE)	A←(DE)	•	•	×	•	×	•	•	•	00 011 010	1A	1	2	7	
LD A,(nn)	A←(nn)	•	•	×	•	×	•	•	•	00 111 010	3A	3	4	13	
										← n →					
										← n →					
LD (BC),A	(BC)←A	•	•	×	•	×	•	•	•	00 000 010	02	1	2	7	
LD (DE),A	(DE)←A	•	•	×	•	×	•	•	•	00 010 010	12	1	2	7	
LD (nn),A	(nn)←A	•	•	×	•	×	•	•	•	00 110 010	32	3	4	13	
										← n →					
										← n →					
LD A,I	A←I	↕	↕	×	0	×	IFF	0	•	11 101 101	ED	2	2	9	
										01 010 111	57				
LD A,R	A←R	↕	↕	×	0	×	IFF	0	•	11 101 101	ED	2	2	9	
										01 011 111	5F				
LD I,A	I←A	•	•	×	•	×	•	•	•	11 101 101	ED	2	2	9	
										01 000 111	47				
LD R,A	R←A	•	•	×	•	×	•	•	•	11 101 101	ED	2	2	9	
										01 001 111	4F				

備考　r，sはレジスタA，B，C，D，E，H，Lを意味します．

IFFは割込み許可フリップ・フロップ(IFF)の内容がP/Vフラグに示されることを意味します．

フラグ表記：　•＝影響受けない，0＝リセットされる，1＝セットされる，×＝不定

↕＝演算結果に従った影響を受ける．

16ビット・ロード命令

ニーモニック	オペレーション	S	Z		H		P/V	N	C	OPコード 76 543 210	Hex	バイト	マシン・サイクル	ステート	コメント
LD dd,nn	dd←nn	•	•	×	•	×	•	•	•	00 dd0 001 ← n → ← n →		3	2	10	dd Pair 00 BC 01 DE 10 HL 11 SP
LD IX,nn	IX←nn	•	•	×	•	×	•	•	•	11 011 101 00 100 001 ← n → ← n →	DD 21	4	4	14	
LD IY,nn	IY←nn	•	•	×	•	×	•	•	•	11 111 101 00 100 001 ← n → ← n →	FD 21	4	4	14	
LD HL,(nn)	H←(nn+1) L←(nn)	•	•	×	•	×	•	•	•	00 101 010 ← n → ← n →	2A	3	5	16	
LD dd,(nn)	dd_H←(nn+1) dd_L←(nn)	•	•	×	•	×	•	•	•	11 101 101 01 dd1 011 ← n → ← n →	ED	4	6	20	
LD IX,(nn)	IX_H←(nn+1) IX_L←(nn)	•	•	×	•	×	•	•	•	11 011 101 00 101 010 ← n → ← n →	DD 2A	4	6	20	
LD IY,(nn)	IY_H←(nn+1) IY_L←(nn)	•	•	×	•	×	•	•	•	11 111 101 00 101 010 ← n → ← n →	FD 2A	4	6	20	
LD (nn),HL	(nn+1)←H (nn)←L	•	•	×	•	×	•	•	•	00 100 010 ← n → ← n →	22	3	5	16	
LD (nn),dd	(nn+1)←dd_H (nn)←dd_L	•	•	×	•	×	•	•	•	11 101 101 01 dd0 011 ← n → ← n →	ED	4	6	20	
LD (nn),IX	(nn+1)←IX_H (nn)←IX_L	•	•	×	•	×	•	•	•	11 011 101 00 100 010 ← n → ← n →	DD 22	4	6	20	
LD (nn),IY	(nn+1)←IY_H (nn)←IY_L	•	•	×	•	×	•	•	•	11 111 101 00 100 010 ← n → ← n →	FD 22	4	6	20	
LD SP,HL	SP←HL	•	•	×	•	×	•	•	•	11 111 001	F9	1	1	6	
LD SP,IX	SP←IX	•	•	×	•	×	•	•	•	11 011 101 11 111 001	DD F9	2	2	10	
LD SP,IY	SP←IY	•	•	×	•	×	•	•	•	11 111 101 11 111 001	FD F9	2	2	10	
PUSH qq	(SP−2)←qq_L (SP−1)←qq_H	•	•	×	•	×	•	•	•	11 qq0 101		1	3	11	qq Pair 00 BC 01 DE 10 HL 11 AF
PUSH IX	(SP−2)←IX_L (SP−1)←IX_H	•	•	×	•	×	•	•	•	11 011 101 11 100 101	DD E5	2	4	15	
PUSH IY	(SP−2)←IY_L (SP−1)←IY_H	•	•	×	•	×	•	•	•	11 111 101 11 100 101	FD E5	2	4	15	
POP qq	qq_H←(SP+1) qq_L←(SP)	•	•	×	•	×	•	•	•	11 qq0 001		1	3	10	
POP IX	IX_H←(SP+1) IX_L←(SP)	•	•	×	•	×	•	•	•	11 011 101 11 100 001	DD E1	2	4	14	
POP IY	IY_H←(SP+1) IY_L←(SP)	•	•	×	•	×	•	•	•	11 111 101 11 100 001	FD E1	2	4	14	

備考　ddはペア・レジスタBC，DE，HL，SPを意味します．

qqはペア・レジスタAF，BC，DE，HLを意味します．

$(ペア・レジスタ)_H$，$(ペア・レジスタ)_L$ は各ペア・レジスタの上位または下位8ビットを意味します．

例：$BC_L = C$，$AF_H = A$

フラグ表記：•＝影響受けない，0＝リセット，1＝セット，×＝不定　↕＝演算結果に従った影響を受ける．

エクスチェンジ命令/ブロック転送命令/ブロックサーチ命令

ニーモニック	オペレーション	フラグ S	Z		H		P/V	N	C	OPコード 76 543 210	Hex	バイト	マシン・サイクル	ステート	コメント
EX DE,HL	DE↔HL	•	•	×	•	×	•	•	•	11 101 011	EB	1	1	4	
EX AF, AF′	AF↔AF′	•	•	×	•	×	•	•	•	00 001 000	08	1	1	4	
EXX	BC↔BC′ DE↔DE′ HL↔HL′	•	•	×	•	×	•	•	•	11 011 001	D9	1	1	4	Register bank and auxiliary register bank exchange
EX (SP),HL	H ↔(SP+1) L ↔(SP)	•	•	×	•	×	•	•	•	11 100 011	E3	1	5	19	
EX (SP),IX	IX_H↔(SP+1) IX_L↔(SP)	•	•	×	•	×	•	•	•	11 011 101 11 100 011	DD E3	2	6	23	
EX (SP),IY	IY_H↔(SP+1) IY_L↔(SP)	•	•	×	•	×	•	•	•	11 111 101 11 100 011	FD E3	2	6	23	
LDI	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1	•	•	×	0	×	①↕	0	•	11 101 101 10 100 000	ED A0	2	4	16	Load(HL) into (DE), increment the pointers and decrement the byte counter(BC)
LDIR	(DE)←(HL) DE←DE+1 HL←HL+1 BC←BC−1 Repeat until BC=0	•	•	×	0	×	0	0	•	11 101 101 10 110 000	ED B0	2 2	5 4	21 16	If BC≠0 If BC=0
LDD	(DE)←(HL) DE←DE−1 HL←HL−1 BC←BC−1	•	•	×	0	×	①↕	0	•	11 101 101 10 101 000	ED A8	2	4	16	
LDDR	(DE)←(HL) DE←DE−1 HL←HL−1 BC←BC−1 Repeat until BC=0	•	•	×	0	×	0	0	•	11 101 101 10 111 000	ED B8	2 2	5 4	21 16	If BC≠0 If BC=0
CPI	A−(HL) HL←HL+1 BC←BC−1	↕	②↕	×	↕	×	①↕	1	•	11 101 101 10 100 001	ED A1	2	4	16	
CPIR	A−(HL) HL←HL+1 BC←BC−1 Repeat until A=(HL) or BC=0	↕	②↕	×	↕	×	①↕	1	•	11 101 101 10 110 001	ED B1	2 2	5 4	21 16	If BC≠0 and A≠(HL) If BC=0 or A=(HL)
CPD	A−(HL) HL←HL−1 BC←BC−1	↕	②↕	×	↕	×	①↕	1	•	11 101 101 10 101 001	ED A9	2	4	16	
CPDR	A−(HL) HL←HL−1 BC←BC−1 Repeat until A=(HL) or BC=0	↕	②↕	×	↕	×	①↕	1	•	11 101 101 10 111 001	ED B9	2 2	5 4	21 16	If BC≠0 and A≠(HL) If BC=0 or A=(HL)

備考 ①もしBC−1=0ならばP/V=0，その他P/V=1

②もしA=(HL)ならばZ=1，その他Z=0

フラグ表記： •=影響受けない，0=リセット，1=セット，×=不定

↕=演算結果に従った影響を受ける.

8ビット算術論理演算命令

ニーモニック	オペレーション	フラグ S	Z		H		P/V	N	C	OPコード 76 543 210	Hex	バイト	マシン・サイクル	ステート	コメント
ADD A, r	A←A+r	↕	↕	×	↕	×	V	0	↕	10 [000] r		1	1	4	r Reg
ADD A, n	A←A+n	↕	↕	×	↕	×	V	0	↕	11 [000] 110		2	2	7	000 B
										← n →					001 C
															010 D
ADD A, (HL)	A←A+(HL)	↕	↕	×	↕	×	V	0	↕	10 [000] 110		1	2	7	011 E
ADD A, (IX+d)	A←A+(IX+d)	↕	↕	×	↕	×	V	0	↕	11 011 101	DD	3	5	19	100 H
										10 [000] 110					101 L
										← d →					111 A
ADD A, (IY+d)	A←A+(IY+d)	↕	↕	×	↕	×	V	0	↕	11 111 101	FD	3	5	19	
										10 [000] 110					以下はADD命令
										← d →					と同様な繰返し
ADC A, s	A←A+s+CY	↕	↕	×	↕	×	V	0	↕	[001]					↓
SUB s	A←A−s	↕	↕	×	↕	×	V	1	↕	[010]					↓
SBC A, s	A←A−s−CY	↕	↕	×	↕	×	V	1	↕	[011]					↓
AND s	A←A∧s	↕	↕	×	1	×	P	0	0	[100]					↓
OR s	A←A∨s	↕	↕	×	0	×	P	0	0	[110]					↓
XOR s	A←A⩛s	↕	↕	×	0	×	P	0	0	[101]					↓
CP s	A−s	↕	↕	×	↕	×	V	1	↕	[111]					↓
INC r	r←r+1	↕	↕	×	↕	×	V	0	•	00 r [100]		1	1	4	INC命令
INC (HL)	(HL)←(HL)+1	↕	↕	×	↕	×	V	0	•	00 110 [100]		1	3	11	INC命令
INC (IX+d)	(IX+d)←(IX+d)+1	↕	↕	×	↕	×	V	0	•	11 011 101	DD	3	6	23	INC命令
										00 110 [100]					
										← d →					
INC (IY+d)	(IY+d)←(IY+d)+1	↕	↕	×	↕	×	V	0	•	11 111 101	FD	3	6	23	INC命令
										00 110 [100]					
										← d →					
DEC s	s←s−1	↕	↕	×	↕	×	V	1	•	[101]					DEC命令はINC命令と同様な繰返し

備考　S＝r，n，(HL)，(IX+d)，(IY+d)

ADD命令と同様な繰返し：[000] ＝ [001] ～ [111]

INC命令→DEC命令＝ [100] → [101]

P/Vフラグ

- 論理演算の場合：　偶数パリティ→P＝1，奇数パリティ→P＝0
- 算術演算の場合：　オーバーフロー有り→V＝1，オーバーフロー無し→V＝0

フラグ表記：　•＝影響受けない，0＝リセット，1＝セット，×＝不定

↕＝演算結果に従った影響を受ける.

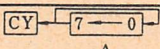
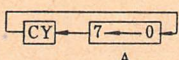
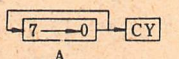
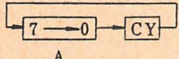
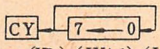
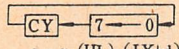
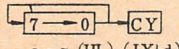
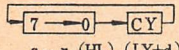
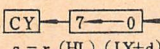
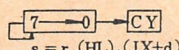
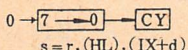
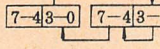
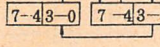
16ビット算術演算命令

ニーモニック	オペレーション	フラグ								OPコード		バイト	マシン・サイクル	ステート	コメント
		S	Z		H		P/V	N	C	76 543 210	Hex				
ADD HL, ss	HL←HL+ss	•	•	×	×	×	•	0	↕	00 ss1 001		1	3	11	ss Reg
															00 BC
ADC HL, ss	HL←HL+ss+CY	↕	↕	×	×	×	V	0	↕	11 101 101	ED	2	4	15	01 DE
										01 ss1 010					10 HL
															11 SP
SBC HL, ss	HL←HL−ss−CY	↕	↕	×	×	×	V	1	↕	11 101 101	ED	2	4	15	
										01 ss0 010					
ADD IX, pp	IX←IX+pp	•	•	×	×	×	•	0	↕	11 011 101	DD	2	4	15	pp Reg
										00 pp1 001					00 BC
															01 DE
															10 IX
															11 SP
ADD IY, rr	IY←IY+rr	•	•	×	×	×	•	0	↕	11 111 101	FD	2	4	15	rr Reg
										00 rr1 001					00 BC
															01 DE
															10 IY
															11 SP
INC ss	ss←ss+1	•	•	×	•	×	•	•	•	00 ss0 011		1	1	6	
INC IX	IX←IX+1	•	•	×	•	×	•	•	•	11 011 101	DD	2	2	10	
										00 100 011	23				
INC IY	IY←IY+1	•	•	×	•	×	•	•	•	11 111 101	FD	2	2	10	
										00 100 011	23				
DEC ss	ss←ss−1	•	•	×	•	×	•	•	•	00 ss1 011		1	1	6	
DEC IX	IX←IX−1	•	•	×	•	×	•	•	•	11 011 101	DD	2	2	10	
										00 101 011	2B				
DEC IY	IY←IY−1	•	•	×	•	×	•	•	•	11 111 101	FD	2	2	10	
										00 101 011	2B				

フラグ表記：　•＝影響受けない，0＝リセット，1＝セット，×＝不定

↕＝演算結果に従った影響を受ける.

アキュームレーター操作命令/CPUコントロール命令

ニーモニック	オペレーション	S	Z		H		P/V	N	C	OPコード 76 543 210	Hex	バイト	マシン・サイクル	ステート	コメント
DAA	Converts Acc, content into packed BCD following add or subtract with packed BCD operands	↕	↕	×	↕	×	P	•	↕	00 100 111	27	1	1	4	Decimal adjust accumulator
CPL	A←$\overline{A}$	•	•	×	1	×	•	1	•	00 101 111	2F	1	1	4	Complement accumulator (1の補数)
NEG	A←$\overline{A}$+1	↕	↕	×	↕	×	V	1	↕	11 101 101 01 000 100	ED 44	2	2	8	Nagate Acc (2の補数)
CCF	CY←$\overline{CY}$	•	•	×	×	×	•	0	↕	00 111 111	3F	1	1	4	Complement carry flag
SCF	CY←1	•	•	×	0	×	•	0	1	00 110 111	37	1	1	4	Set carry flag
NOP	No operation	•	•	×	•	×	•	•	•	00 000 000	00	1	1	4	
HALT	CPU halted	•	•	×	•	×	•	•	•	01 110 110	76	1	1	4	
DI	IFF←0	•	•	×	•	×	•	•	•	11 110 011	F3	1	1	4	
EI	IFF←1	•	•	×	•	×	•	•	•	11 111 011	FB	1	1	4	
IM 0	Set interrupt mode 0	•	•	×	•	×	•	•	•	11 101 101 01 000 110	ED 46	2	2	8	
IM 1	Set interrupt mode 1	•	•	×	•	×	•	•	•	11 101 101 01 010 110	ED 56	2	2	8	
IM 2	Set interrupt mode 2	•	•	×	•	×	•	•	•	11 101 101 01 011 110	ED 5E	2	2	8	

備考　IFF＝割込イネーブル・フリップ・フロップ

CY　＝キャリー・フリップ・ブロップ

フラグ表記：　•＝影響受けない，0＝リセット，×＝不定

↕＝演算結果に従った影響を受ける.

ローテート・シフト命令

ニーモニック	オペレーション	フラグ S	Z		H		P/V	N	C	OPコード 76 543 210	Hex	バイト	マシン・サイクル	ステート	コメント
RLCA	CY ← [7 ← 0] (A)	•	•	×	0	×	•	0	↕	00 000 111	07	1	1	4	Rotate left circular accumulator
RLA	[CY ← 7 ← 0] (A)	•	•	×	0	×	•	0	↕	00 010 111	17	1	1	4	Rotate left accumulator
RRCA	[7 → 0] → CY (A)	•	•	×	0	×	•	0	↕	00 001 111	0F	1	1	4	Rotate right circular accumulator
RRA	[7 → 0 → CY] (A)	•	•	×	0	×	•	0	↕	00 011 111	1F	1	1	4	Rotate right accumulator
RLC r	CY ← [7 ← 0] r,(HL),(IX+d),(IY+d)	↕	↕	×	0	×	P	0	↕	11 001 011 00 [000] r	CB	2	2	8	Rotate left circular ragister r
RLC (HL)		↕	↕	×	0	×	P	0	↕	11 001 011 00 [000] 110	CB	2	4	15	r Reg 000 B 001 C
RLC (IX+d)		↕	↕	×	0	×	P	0	↕	11 011 101 11 001 011 ← d → 00 [000] 110	DD CB	4	6	23	010 D 011 E 100 H 101 L 111 A
RLC (IY+d)		↕	↕	×	0	×	P	0	↕	11 111 101 11 001 011 ← d → 00 [000] 110	FD CB	4	6	23	
RL s	[CY ← 7 ← 0] s≡r,(HL),(IX+d),(IY+d)	↕	↕	×	0	×	P	0	↕	[010]					
RRC s	[7 → 0] → CY s≡r,(HL),(IX+d),(IY+d)	↕	↕	×	0	×	P	0	↕	[001]					
RR s	[7 → 0 → CY] s≡r,(HL),(IX+d),(IY+d)	↕	↕	×	0	×	P	0	↕	[011]					
SLA s	CY ← [7 ← 0] ← 0 s≡r,(HL),(IX+d),(IY+d)	↕	↕	×	0	×	P	0	↕	[100]					
SRA s	[7 → 0] → CY s≡r,(HL),(IX+d),(IY+d)	↕	↕	×	0	×	P	0	↕	[101]					
SRL s	0 → [7 → 0] → CY s≡r,(HL),(IX+d),(IY+d)	↕	↕	×	0	×	P	0	↕	[111]					
RLD	A [7-4│3-0] [7-4│3-0] (HL)	↕	↕	×	0	×	P	0	•	11 101 101 01 101 111	ED 6F	2	5	18	
RRD	A [7-4│3-0] [7-4│3-0] (HL)	↕	↕	×	0	×	P	0	•	11 101 101 01 100 111	ED 67	2	5	18	

フラグ表記：　• = 影響受けない，0 = リセット，1 = セット，× = 不定

↕ = 演算結果に影響を受ける.

ビット操作命令

ニーモニック	オペレーション	S	Z		H		P/V	N	C	OPコード 76 543 210	Hex	バイト	マシンサイクル	ステート	コメント
BIT b, r	$Z \leftarrow \overline{r}_b$	×	↕	×	1	×	×	0	•	11 001 011	CB	2	2	8	r | Reg
										01 b r					000 | B
BIT b, (HL)	$Z \leftarrow \overline{(HL)}_b$	×	↕	×	1	×	×	0	•	11 001 011	CB	2	3	12	001 | C
										01 b 110					010 | D
BIT b, (IX+d)	$Z \leftarrow \overline{(IX+d)}_b$	×	↕	×	1	×	×	0	•	11 011 101	DD	4	5	20	011 | E
										11 001 011	CB				100 | H
										← d →					101 | L
										01 b 110					111 | A
															b | Bit Tested
BIT b, (IY+d)	$Z \leftarrow \overline{(IY+d)}_b$	×	↕	×	1	×	×	0	•	11 111 101	FD	4	5	20	000 | 0
										11 001 011	CB				001 | 1
										← d →					010 | 2
										01 b 110					011 | 3
															100 | 4
															101 | 5
															110 | 6
															111 | 7
SET b, r	$r_b \leftarrow 1$	•	•	×	•	×	•	•	•	11 001 011	CB	2	2	8	
										11 b r					
SET b, (HL)	$(HL)_b \leftarrow 1$	•	•	×	•	×	•	•	•	11 001 011	CB	2	4	15	
										11 b 110					
SET b, (IX+d)	$(IX+d)_b \leftarrow 1$	•	•	×	•	×	•	•	•	11 011 101	DD	4	6	23	SET命令
										11 001 011	CB				
										← d →					
										11 b 110					
SET b, (IY+d)	$(IY+d)_b \leftarrow 1$	•	•	×	•	×	•	•	•	11 111 101	FD	4	6	23	
										11 001 011	CB				
										← d →					
										11 b 110					
RES b, s	$s_b \leftarrow 0$									10					RES命令はSET
	s≡r, (HL),														命令と同様な繰返し
	(IX+d),														
	(IY+d)														

備考　SET命令→RES命令＝ 11 → 00

フラグ表記：　• ＝影響受けない，0＝リセット，1＝セット，×＝不定

↕＝演算結果に従った影響を受ける.

ジャンプ命令

ニーモニック	オペレーション	フラグ S	Z		H		P/V	N	C	OPコード 76 543 210	Hex	バイト	マシン・サイクル	ステート	コメント
JP nn	PC←nn	•	•	×	•	×	•	•	•	11 000 011	C3	3	3	10	
										← n →					
										← n →					cc 条件
JP cc,nn	If condition cc	•	•	×	•	×	•	•	•	11 cc 010		3	3	10	000 NZ non zero
	is true PC←nn,									← n →					001 Z zero
	otherwise									← n →					010 NC non cary
	continue														011 C carry
															100 PO parity odd
															101 PE parity even
															110 P sign positive
JR e	PC←PC+e	•	•	×	•	×	•	•	•	00 011 000	18	2	3	12	111 M sign negative
										← e−2 →					
JR C,e	If C=0,	•	•	×	•	×	•	•	•	00 111 000	38	2	2	7	
	continue									← e−2 →					
	If C=1,											2	3	12	
	PC←PC+e														
JR NC,c	If C=1,	•	•	×	•	×	•	•	•	00 110 000	30	2	2	7	
	continue									← e−2 →					
	If C=0,											2	3	12	
	PC←PC+e														
JR Z,e	If Z=0	•	•	×	•	×	•	•	•	00 101 000	28	2	2	7	
	continue									← e−2 →					
	If Z=1,											2	3	12	
	PC←PC+e														
JR NZ,e	If Z=1,	•	•	×	•	×	•	•	•	00 100 000	20	2	2	7	
	continue									← e−2 →					
	If Z=0											2	3	12	
	PC←PC+e														
JP (HL)	PC←HL	•	•	×	•	×	•	•	•	11 101 001	E9	1	1	4	
JP (IX)	PC←IX	•	•	×	•	×	•	•	•	11 011 101	DD	2	2	8	
										11 101 001	E9				
JP (IY)	PC←IY	•	•	×	•	×	•	•	•	11 111 101	FD	2	2	8	
										11 101 001	E9				
DJNZ e	B←B−1	•	•	×	•	×	•	•	•	00 010 000	10	2	2	8	
	If B=0,									← e−2 →					
	continue														
	If B≠0,											2	3	13	
	PC←PC+e														

備考　e＝レラティブ・アトレッシング・モードにおける変位値

（符号付き２の補数＝＋１２７～－１２８）

e－２＝eの実効変位置（レラティブ・アドレッシングの項参照）

フラグ表記：　•＝影響受けない，０＝リセット，１＝セット，×＝不定

↕＝演算結果に従った影響を受ける．

コール命令/リターン命令

ニーモニック	オペレーション	S	Z		H		P/V	N	C	76 543 210	Hex	バイト	マシン・サイクル	ステート	コメント
CALL nn	(SP-1)←PC_H	•	•	×	•	×	•	•	•	11 001 101	CD	3	5	17	
	(SP-2)←PC_L									← n →					
	PC←nn									← n →					
CALL cc,nn	If condition	•	•	×	•	×	•	•	•	11 cc 100		3	3	10	
	cc is false									← n →					
	continue,									← n →		3	5	17	
	otherwise														
	same as														
	CALL nn														
RET	PC_L←(SP)	•	•	×	•	×	•	•	•	11 001 001	C9	1	3	10	
	PC_H←(SP+1)														
RET cc	If condition	•	•	×	•	×	•	•	•	11 cc 000		1	1	5	
	cc is false														
	continue,											1	3	11	
	otherwise														
	same as														
	RET														
RETI	Return from	•	•	×	•	×	•	•	•	11 101 101	ED	2	4	14	
	interrupt									01 001 101	4D				
RETN	Return from	•	•	×	•	×	•	•	•	11 101 101	ED	2	4	14	
	non maskable									01 000 101	45				
	interrupt														
RST p	(SP-1)←PC_H	•	•	×	•	×	•	•	•	11 t 111		1	3	11	
	(SP-2)←PC_L														
	PC_H←0														
	PC_L←p														

cc	条件	
000	NZ	non zero
001	Z	zero
010	NC	non carry
011	C	carry
100	PO	parity odd
101	PE	parity even
110	P	sign positive
111	M	sign negative

t	p
000	00H
001	08H
010	10H
011	18H
100	20H
101	28H
110	30H
111	38H

フラグ表記： •＝影響受けない，0＝リセット，1＝セット，×＝不定

↕＝演算結果に従った影響を受ける.

入出力命令

ニーモニック	オペレーション	S	Z		H		P/V	N	C	OPコード 76 543 210	Hex	バイト	マシン・サイクル	ステート	コメント
IN A,n	A←(n)	•	•	×	•	×	•	•	•	11 011 011 ← n →	DB	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
IN r,(C)	r←(C) if r=110 only the flags will be affected	↕	↕	×	↕	×	P	0	•	11 101 101 01 r 000	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INI	(HL)←(C) B←B−1 HL←HL+1	×	③ ↕	×	×	×	×	1	•	11 101 101 10 100 010	ED A2	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INIR	(HL)←(C) B←B−1 HL←HL+1 Repeat until B=0	×	1	×	×	×	×	1	•	11 101 101 10 110 010	ED B2	2 2	5 (If B≠0) 4 (If B=0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
IND	(HL)←(C) B←B−1 HL←HL−1	×	③ ↕	×	×	×	×	1	•	11 101 101 10 101 010	ED AA	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
INDR	(HL)←(C) B←B−1 HL←HL−1 Repeat until B=0	×	1	×	×	×	×	1	•	11 101 101 10 111 010	ED BA	2 2	5 (If B≠0) 4 (If B=0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUT n,A	(n)←A	•	•	×	•	×	•	•	•	11 010 011 ← n →	D3	2	3	11	n to $A_0 \sim A_7$ Acc to $A_8 \sim A_{15}$
OUT (C),r	(C)←r	•	•	×	•	×	•	•	•	11 101 101 01 r 001	ED	2	3	12	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTI	(C)←(HL) B←B−1 HL←HL+1	×	③ ↕	×	×	×	×	1	•	11 101 101 10 100 011	ED A3	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTIR	(C)←(HL) B←B−1 HL←HL+1 Repeat until B=0	×	1	×	×	×	×	1	•	11 101 101 10 110 011	ED B3	2 2	5 (If B≠0) 4 (If B=0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OUTD	(C)←(HL) B←B−1 HL←HL−1	×	③ ↕	×	×	×	×	1	•	11 101 101 10 101 011	ED AB	2	4	16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$
OTDR	(C)←(HL) B←B−1 HL←HL−1 Repeat until B=0	×	1	×	×	×	×	1	•	11 101 101 10 111 011	ED BB	2 2	5 (If B≠0) 4 (If B=0)	21 16	C to $A_0 \sim A_7$ B to $A_8 \sim A_{15}$

備考　③もしBC−1=0ならばZ=1，その他Z=0

フラグ表記：　•=影響受けない．0=リセット，1=セット，×=不定

↕=演算結果に従った影響を受ける．

教育用 ソフトテープシリーズ

きょうから マイコンがあなたの家庭教師

☆マイ単語レッスン

開発：DATA POP

プログラム名	内容	MZ 80B 2000		FM−7／8		PC−6001 (32K)		PC−8001	
		コードナンバー	定価	コードナンバー	定価	コードナンバー	定価	コードナンバー	定価
中学１年生用	①必須英単語180語②連語・短文20～60③アクセントの問題④発声練習（本邦初の画面の単語を見ながらテープに録音してある米人講師の発音に合わせてレッスンする）〈パッケージ内容〉マニュアル……１冊、ソフトテープ……１巻、発音練習テープ……１巻 ※言語BASIC	3382	3,700円	3000	3,700円	3050	3,400円	1988	3,700円
中学２年生用		3383	4,100円	3001	4,100円	3052	3,800円	1996	4,100円
中学３年生用		3384	4,100円	3002	4,100円	3052	3,800円	1997	4,100円
高校受験用		3385	4,100円	3018	4,100円	3064	3,800円	3764	4,100円
一般用			開発中		開発中		開発中		開発中

☆中学数学マスターシリーズ全13巻

開発：DATA POP

	プログラム名	内容	FM−7／8		PC−6001		PC−8001	
			コードナンバー	定価	コードナンバー	定価	コードナンバー	定価
❶	連立方程式	例題で加減法、代入法をマスターし、整数解、分数解を選択	1287	3,000円	※1028	2,700円	1974	3,000円
❷	図形の１	平行線と三角形の性質＝同位角、錯角、対頂角と三角形の内角の和と性質を学習。	1288	3,000円	1029	2,700円	1975	3,000円
❸	式の展開	多項式の展開＝カッコの計算と同種類の整理を学習。	1289	3,500円	※1030	2,700円	1976	3,500円
❹	一式関数とグラフ(1)	座標とグラフ＝比例と反比例点の座標とグラフでの対応関係をマスター。	1290	3,500円	※1031	2,700円		
❺	一次関数とグラフ(2)	一次方程式とグラフ＝直線の方程式を理解したうえで、一次方程式の解を求める。	1291	3,500円	※1032	2,700円		
❻	一次関数とグラフ(3)	一次不等式とグラフ＝直線の方程式を用いて領域(定義域)を表示、グラフから不等式を理解。	1292	3,500円	※1033	2,700円		
❼	一次方程式	さまざな未知数の方程式を解くことで変数になれ、中学数学の関門を突破する。	1293	3,000円	※1034	2,700円	1980	3,000円
❽	一次方程式	一元＝直線上での定義域、領域をマスター。グラフによる解法を確固する。	1294	3,000円	※1035	2,700円	1981	3,000円
❾	図形の２	面積計算＝三角形、四角形、円の組合わせからできる図形の面積を求める。	1295	3,000円	※1036	2,700円		
❿	最大公約数と最小公倍数	因数分解など式の整理の準備	1296	3,000円	※1037	2,700円	1983	3,000円
⓫	因数分解	ランク１：Xの係数は１　ランク２：Xの係数は１とはかぎらない	1297	3,000円	※1038	2,700円	1984	3,000円
⓬	二次方程式	ランク１：Xの係数は１　ランク２：Xの係数は１とはかぎらない	1298	3,000円	※1039	2,700円	1985	3,000円
⓭	統計とグラフ	ヒストグラムにより統計を理解する	1299	3,000円	※1040	2,700円	※1986	3,000円

※は32K

☆英作文シリーズ(EXPRT ENGLISH)全15巻

開発：数研塾

プログラム名	内容	PC−6001		PC−8001	
		コードナンバー	定価	コードナンバー	定価
中学１年生用	PART１：Be動詞を使った文。What、Whoを使った文。 PART２：haveを使った命令形。 PART３：I、You文。複数形。Which文。 PART４：助動詞、When、how文。 PART５：進行形。there、when、can 文。	3053	全５巻 15,000円	3752	全５巻 15,000円
中学２年生用	PART１：過去形。 PART２：比較文。 PART３：未来形。 PART４：不定詞、受動態。 PART５：２年総復習。	3054	全５巻 15,000円	3753	全５巻 15,000円
中学３年生用	PART１：動名詞。 PART２：現在完了形の応用。受動態の応用。 PART３：不定詞、間接疑問文、現在分詞、過去分詞。 PART４：関係代名詞。 PART５：中学総復習。	3055	全５巻 15,000円	3754	全５巻 15,000円

☆中学社会科シリーズ　各3,000円

学習コースで年号や地名をマスター。問題コースで３分間テストにトライしよう。マイコンが採点しまちがった答は正解を教えてくれます。塾で実際に使われているソフトだから、実戦的で成績アップにつながります。

品名／コード ＼ 機種	PC−6001 (32K)	PC−8001 (32K)	MZ 700／1200 共通	FM-7／8 共通
日本史	1013	3822	3555	3952
世界史	1014	3823	3556	3953
日本地理	1015	3824	3557	3954
世界地理	1016	3825	3558	3955

◘教育用ソフトテープのお申込みは下記へ……

①DEMPAマイコンソフト全国取扱店　②マイコンショップ　③書　店　④電波新聞社全国各支局

電波新聞社　〒141 東京都品川区東五反田1-11-15　電話(03)445-6111

大阪本社 06-203-3361　札幌支局 011-641-5591　静岡支局 0542-54-6405　金沢支局 0762-63-8661　広島支局 082-228-5581　熊本支局 0963-80-7500

西部本社 092-431-7411　仙台支局 0222-27-7211　新潟支局 0252-45-2526　京都支局 075-221-8021　高松支局 0878-61-3111　鹿児島支局 0992-26-3630

名古屋支局 052-261-4541　関東総局 0273-26-3206　松本支局 0263-36-0445　神戸支局 078-391-5885　下関支局 0832-67-7478

☆算数学習

開発：オーク

	プログラム名	内容	MZ−80B 2000 コードナンバー	MZ−80B 2000 定価	FM−8 コードナンバー	FM−8 定価	PC−6001(32K) コードナンバー	PC−6001(32K) 定価	PC−8001(32K) コードナンバー	PC−8001(32K) 定価	PC−8801 コードナンバー	PC−8801 定価
計算編	たし算	整数の加法(たしざん)25レベル(内筆算は13レベルから)	3374	3,600円	3003	3,600円	3056	3,200円	3755	3,600円	3601	3,600円
	ひき算	整数の減法(ひきざん)25レベル(内筆算は13レベルから)	3375	3,600円	3004	3,600円	3057	3,200円	3756	3,600円	3602	3,600円
	かけ算	整数の乗法(かけざん)12レベル(内筆算は7レベルから)	3376	3,600円	3005	3,600円	3058	3,200円	3757	3,600円	3603	3,600円
	わり算	整数の除算(わりざん)16レベル(内筆算は8レベルから)	3377	3,600円	3006	3,600円	3059	3,200円	3758	3,600円	3604	3,600円
	分数1	13レベル	3378	3,600円	3007	3,600円	3060	3,200円	3759	3,600円	3605	3,600円
	分数2	5レベル	3379	3,600円	3008	3,600円	3061	3,200円	3760	3,600円	3606	3,600円
	小数1	10レベル	3380	3,600円	3009	3,600円	3062	3,200円	3761	3,600円	3607	3,600円
	小数2	5レベル	3381	3,600円	3010	3,600円	3063	3,200円	3762	3,600円	3608	3,600円
図形編	図形1年	1年「ボールあそび」の絵から、(まる)や(さんかく)、(しかく)などの形の勉強をします。			3011	3,600円					3609	3,600円
	図形2年	2年位置の表わし方、直角、三角形、四角形、直角三角形、長方形、正方形、長さの測り方。			3012	3,600円					3610	3,600円
	図形3年	3年円(中心、直径、半径、円のいろいろな性質)、二等辺三角形、正三角形。			3013	3,600円					3611	3,600円
	図形4年①	4年角、角の測り方、角の計算、平行、垂直、面積、のいろいろな単位、四角形の分類、直方体、立方体、空間の直線の平行、体積、体積の単位			3014	3,600円					3612	3,600円
	図形4年②				3015	3,600円					3613	3,600円
	図形5年	5年垂線、垂直二等分線、線対称、点対称、距離の最大最小、面積の求め方、等積変形、おうぎ形、いろいろな形の面積。			3016	3,600円					3614	3,600円
	図形6年	6年相似、相似比、空間の位置の表わし方、面の平行と垂直、ねじれ、投影図、展開図、角柱、円柱、角すい、円すい、これらの表面積、体積。			3017	3,600円					3615	3,600円

FM-8，PC-8801は漢字ROMが必要です。

☆いんぷっと学習シリーズ for PC-8001

開発：日本マイコン学院

	プログラム名	内容	PC−8001(32K) コードナンバー	PC−8001(32K) 定価
❶	基本作図と条件を満たす点	「条件を満たす点の集合」と「基本作図」に関して解説。作図の手順を印象強く示し、やや高度な応用問題がヒントと解答つきで紹介される。	3787	3,600円
❷	面積公式とその応用	三角形・四角形・円(扇形)の公式と解説。これらに関しての面積を求める基本・応用の問題を含む。	3788	3,600円
❸	体積・表面積の公式とその応用	角柱、円柱、角錐円錐等の立体の体積・表面積の公式の解説。これらの体積・表面積を求める問題を含む。	3789	3,600円
❹	平行線の性質・対頂角	直線のなす角から対頂角・平行線の錯角・同位角・同側内角を解説する。それらに関する基本問題から応用問題を含む。	3790	3,600円
❺	三角形・四角形・多角形の内角・外角	三角形・四角形の内角と外角・多角形の対角線の数と内角の和等を図と公式で解説。これらについての問題を含む。	3791	3,600円
❻	証明問題の基礎(三角形)	三角形(直角三角形含む)の合同条件を明示し、代表的な証明問題を含む。	3792	3,600円
❼	証明問題の基礎(平行四辺形)	平行四辺形の性質、平行四辺形になるための条件を解説。これらを利用した代表的証明問題を含む。	3793	3,600円
❽	平行線と線分の比	平行線と線分の間の長さに関する比の定理を図で解説。これらについて長さを求める問題を含む。	3794	3,600円
❾	三角形の相似	三角形の相似条件を解説。相似の判定、長さを求める問題。証明問題を含む。	3795	3,600円
❿	中点連結定理・内心外心重心	三角形・台形の中点連結定理とその練習。重心・外心・内心の基本解説。	3796	3,600円
⓫	円(円周角・接弦定理・内接四角形)	円周角・円に内接する四角形、接弦定理をわかりやすく解説。	3797	3,600円
⓬	1年英単語(プリンス)	中学生の教科書にあわせて単語や連語を紹介する。	3798	3,600円
⓭	2年英単語(プリンス)	中学生の教科書にあわせて単語や連語を紹介する。	3799	3,600円
⓮	3年英単語(プリンス)	中学生の教科書にあわせて単語や連語を紹介する。	3800	3,600円
⓯	1年英単語(ホライズン)	中学生の教科書にあわせて単語や連語を紹介する。	3801	3,600円
⓰	2年英単語(ホライズン)	中学生の教科書にあわせて単語や連語を紹介する。	3802	3,600円
⓱	3年英単語(ホライズン)	中学生の教科書にあわせて単語や連語を紹介する。	3803	3,600円
⓲	動詞の活用	動詞の変化を効率よく学習するプログラム。規則動詞は語尾の発音・不規則動詞は変化の類似点による分類そしてハイスコアー形式の実力テストなどを含む。	3804	3,600円
⓳	同音異義語・反意語	高校入試によく出題される同音異義語、特殊な複数形、反意語(動詞・名詞・形容詞その他に分類)を学習するプログラム。単語の紹介、練習テスト・実力テストなどを含む	3805	3,600円
⓴	発音問題	色々なパターンで発音問題を基礎から応用までとらえたプログラム。高校入試には必須。アクセント問題を含む。	3806	3,600円
㉑	前置詞	連語・時・場所・その他に関する前置詞の意味と用法を例文つきで解説し、虫喰い問題や単語のならべかえによる英文完成問題などを含む。	3807	3,600円
㉒	接続詞	各種接続詞の意味と用法を例文をつけて解説し、虫喰い問題や英文完成問題(単語のならびかえ)などで学力の定着をはかる。	3808	3,600円
㉓	助動詞	助動詞の用法を意味・同意表現・時制・疑問文・否定文まで例文をつけて解説する。ハイスコアー形式の虫喰い問題や英文完成(単語のならびかえ)問題を含む。	3809	3,600円
㉔	必修連語	特に重要な連語を順次紹介する。語句の練習テストの他に虫喰い問題や単語のならびかえによる英文完成問題を多数含む	3810	3,600円
㉕	力のつりあい・合成	力についてわかりやすい図によって解説する。力の矢印の書き方も明解に説明。てこ、ばね等力の大きさ、2力の合成、3力のつりあい問題を含む	3811	3,600円
㉖	斜面・仕事量	斜面上の物体に働く力のつりあいと、物体を引き上げる仕事と仕事率の解説。	3812	3,600円
㉗	水圧器・大気圧	圧力、水圧をわかりやすく図で解説。ハスカルの原理を油圧器で解説しながら練習する。	3813	3,600円
㉘	オームの法則・抵抗	電流、電圧、抵抗の関係をグラフと回路図で解説。オームの法則と合成抵抗の練習問題を含む。	3814	3,600円
㉙	電力・ジュールの法則	オームの法則、電力、発熱量のわかりやすい図解と表による解説。	3815	3,600円
㉚	原子記号	主な原子記号を紹介する。(索引付)。練習は原子記号又は原子名を解答する。	3816	3,600円
㉛	化学式(分子式)	化合物の原子の組み合わせを理解しやすい原子価で解説。	3817	3,600円
㉜	化学反応式	中学生のための主な化学反応式を表示します。化学反応式の基本的な問題を含む。	3818	3,600円

新作ゲームコーナー

テープ名	内容	定価	機種名	言語	コードナンバー	注文数
MZ-700用						
ゾンビ・パニック	レベルが5段階。低レベルが終ったら高レベルへ挑戦して下さい。ゾンビがあなたを襲ってくる。	3,200円	MZ-700	M	3515	
ダイヤモンドコブラ	制限時間内に10個のダイヤモンドを獲得するゲーム。各部屋に1個のダイヤモンド、しかし、多くのコブラが……	3,200円	MZ-700	M	3516	
スーパーゴルフ	ホールインワンを目指して頑張れ！方向をまちがえると……。	3,800円	MZ-700		3517	
ムーンサルト	ロケットの高さとスピードをうまくコントロールして着陸して下さい。	2,800円	MZ-700	B	3518	
カンニング大作戦	新米教師のあなた、試験中にカンニングしている学生が多勢います。	2,800円	MZ-700	B	3519	
緊急発進	航空機のフライトシュミレーションと攻撃ゲームを合わせた新しいゲームあなたは操縦士、離着陸に失敗しないように！	2,800円	MZ-700	B	3520	
ホース・レース	単勝で簡単に楽しめる競馬ゲーム、さああなたは何番に賭ける。	3,200円	MZ-700	HuB	3521	
エアーバトル	君はパイロット。襲いかかるＵＦＯをやっつけろ！ＵＦＯからのミサイルはランダムにおそいかかる。	2,800円	MZ-700	M	3522	
HUPACK ＃1	マージャンと雀球が一つにコンパクトになった！	3,200円	MZ-700	HuB	3523	
HUPACK ＃2	ローディボールと野球拳が一つにコンパクトになった。	3,200円	MZ-700	HuB	3524	
HUPACK ＃3	アリゲータとパチンコが一つにコンパクトになった。	3,200円	MZ-700	HuB	3525	
HUPACK ＃4	スタートレックとレイホープ研究所が一つにコンパクトになった！	3,200円	MZ-700	HuB	3526	
HUPACK ＃5	ノルマンディーとラスベガススロットが一つにコンパクトになった！	3,200円	MZ-700	HuB	3527	
MZ-2000用						
夢のプロ野球	巨人対西武戦。あはたは監督。サインを出しての攻防戦。	3,200円	MZ-2000	B	3426	
X-1用						
レーザープラネット	岩だらけの狭いこの宇宙道。危い！岩かげから機雷が……。素早いキー操作でこの危機を脱出だ！	3,000円	X-1	M	3240	
フルーツサーキット	フルーツバスケットが空から落ちてきた。道路にはバナナ、リンゴ、チェリーがいっぱい。さあ早い者勝ちだ！	3,000円	X-1	M	3241	
MZ-700用						
プロ野球	あなたはプロ野球の監督だ！チームを勝利へ導いて下さい。2人で燃える野球ゲーム。	3,000円	MZ-700	M	3531	
バクテリア	突然出現した猛毒バクテリア。容器を破らないように退治せよ！	3,000円	MZ-700	M	3530	
実務新作ソフトコーナー						
HUCAL	最高級ビジネス用簡易言語。ビジネスなどで使用する合計に便利。平均は内部関数で処理。横254×縦はメモリー限界まで表作成可能。編集機能付。	9,800円	80B	M		
MZ-700用						
ムービング・サーチャー	追いかけてくるエイリアン、飛びかうボールをよけ、"$"マークを拾い集めるゲーム。百万ドルを奪い去れ！	2,800円	MZ-700	M	3532	
バトルゲーム（2人用）	二人で、お互に妨害しながらボールを打ち合い、相手のゴールに入れて下さい。玉も2つ、3つと増えてきます。	2,800円	MZ-700	M	3533	
ザ・ラストファイター（2人用）	フェザー・トーピードを相手にあてると相手のエネルギーが減ります。相手のエネルギーが0になると勝ちです。	2,800円	MZ-700	M	3534	
スペース・ファイター	UFOの攻撃から宇宙基地を守るのが、あなたの使命。撃って、撃って、撃ちまくれ！	2,800円	MZ-700	M	3535	
ボート対コンピューター	コンピューターと競艇、コンピューターはチームワークでだんだん強くなってきます。	2,800円	MZ-700	M	3536	
バリケード（2人用）	ひと味違ったバリケードゲーム。自分の足跡に相手がぶつかると勝ち。	2,800円	MZ-700	M	3537	
タンクバトル（2人用）	それぞれの戦手の戦車を、必中ミサイルで撃破。障壁に向って直角にミサイルを発射すると、はねかえり自爆します。	2,800円	MZ-700	M	3538	
イレイザー対デバッガー	イレイザーをたくみに操作して、デバッガーにつかまらないように素早くドット（点）を消して下さい。	2,800円	MZ-700	M	3539	
ボンバーズパニック	BOMBERをつかまえるゲーム。BOMBERは爆発するので、爆発するまえに、はやくつかまえて下さい。	2,800円	MZ-700	M	3542	
スネーク対エイリアン	スネークを操作して、うろうろしているエイリアンを食べて下さい。エイリアンを食べると、しっぽが伸びます。	2,800円	MZ-700	M	3545	
ブレイクアウト・ストライクバック	画面の右下にブロックが置いてあります。ボールに当らないようにブロックを積み上げるゲーム。	2,800円	MZ-700	M	3546	
アラート・ゾーン	エイリアンをさけながらアルファベットをAからZまで順番に食べていくゲーム。	2,800円	MZ-700	M	3547	
シールド・ゾーン	ブロックやボールをよけながら、タイムが0になるまで逃げます。下手をするとどんどん壁が作られます。	2,800円	MZ-700	M	3548	
キャノン・ファイター	左右に流れる星の間をぬって、上にいるコンピューターの砲台を破壊するゲーム。	2,800円	MZ-700	M	3549	
ドラキュラ・パニック	ドラキュラにつかまらないようにドラキュラ城を破壊して下さい。戸びらが開ききったとき十字架ブーメランで破壊できます。	2,800円	MZ-700	M	3550	
キャッチング・フェアー	島と島の間を飛びはねるノミをつぶすゲーム。だんだんとノミのスピードが速くなります。	2,800円	MZ-700	M	3551	
ワーム・ゾーン	果てしなく襲来するＵＦＯとワームに対する人類最後の挑戦が始まった。残された人類のただ一つの希望はロボットHHP－9000	2,800円	MZ-700	M	3540	
ダイヤモンド・チェイス	左側のレーダーを見て、迷路に住むモンスターに注意し、かくされたダイヤモンド10個をひろい集め、出口に逃げ出すゲーム。	2,800円	MZ-700	M	3552	
ブリックス	クラブマークをさけながら、ボールを上下、左右、斜めに動かし、アルファベットをAから順番に食べていくゲーム。	2,800円	MZ-700	M	3541	
MZ-1200用						
バトルゲームII（2人用）	2人でお互に妨害しながら飛びかうボールを相手のゴールに入れて下さい。	2,800円	MZ-1200	M		
X-1用						
ファンタジックキュービック PART I	魅惑の立方体、君はこの謎に何段階まで挑戦できるか？	3,000円	X-1	B	3242	

マイコンソフトテープ一覧表

送料
1本……………200円
2本以上1本増すごとに100円増
5本以上は送料無料

テープ名	内容	定価	機種名	言語	コードナンバー	注文数
月刊マイコン オリジナル・ソフト	月刊マイコンに記事掲載された、他にないユニークなプログラムです。					
リアルタイム SUPER STAR TREK	今までのTREKゲームの常識をうち破った傑作。ワープ航法、長距離レーダー始動防御スクリーン作動、積載コンピュータをフル活用して、クリゴンと頭脳戦争だ！	3,000円	PC-8001(32K) / MICRO-8	B	1735 / 3032	
みみずの滝のぼり	迫りくるゲジゲジの大群に果敢にたち向かうミミズの勇士。でもゲジゲジにつかまると、ゲジゲジが次々と成長し状況悪化。ゆけミミズ戦士よボーナスの日まで！	3,000円	PC-8001(32K)	B	1736	
コードネーム自動表示	ピアノ・ギター楽譜のコード進行チャート、コード修正をスピーディに！ピアノとギターが同時に表示され、またコードを楽譜化して見ることが出来ます。	3,000円	PC-8001(32K)	B	1737	
インデアン・ポーカー	PCとあなたのしのぎを削る賭け金の競い合い。強気になったり、弱気になったり、いかにも人間らしくふるまうPC。あなたとPC、どちらが破産？	3,000円	PC-8001(32K)	M B	1738	
SUPER卓球ゲーム	本物そっくりの卓球ゲーム。ラケットスイングができ、打球角度を自由にコントロールできます。コンピュータ相手にパーフェクト試合ができたらあなたは天才！	3,000円	PC-8001(32K)	M B	1739	
エイリアンビリヤード	エイリアン相手にビリヤード！あなたのたくみなスティックさばきで見事にエイリアンを撃退してください。マシン語&BASICオートスタートです。	3,000円	PC-8001(32K)	M B	1740	
少年とエイリアン	宇宙元年8001年、月面基地に生き残った少年3人と異星人との激しい戦い。勝ち残った少年だけが、栄光のエイリアンレースに参加できます。	3,000円	PC-8001(32K)	M B	1742	
成績処理	①集計表（合計、平均、標準偏差）、②ヒストグラム各種、③素点表（順位、偏差値を含む）、④偏差値表（各教科の偏差値とその散らばり）、⑤個人向けカード、⑥順位表以上の処理が出来ます。1クラス45名で最大7クラスまで可能。	3,000円	PC-8001(32K) / MICRO-8	B	1743 / 3034	
ピラミッドとミイラ	オセロとルービックキューブを組合せた様なゲームで、系統的に考えていかないとなかなか完成しません。たとえ完成出来ずGIVE UPしてもあとはPC—8001が考えて完成させてくれます。	3,000円	PC-8001(32K)	M B	1790	
ALIAN LAND	人類の平和を守るため、ロボットをうまく操縦して下さい。エイリアンを避けて、エネルギー鉱石を一つでも多く取って下さい。アタックエイリアンに要注意。	3,000円	PC-8001(32K)	M B	1957	
スーパー・ムービング・ブロック	ワーピング・ラケット、攻撃するエイリアン、天じゃステーションが笑っている。ワザ有り、運有り、度胸有り、オールマシン語でスピードも抜群。	3,000円	PC-8001(32K)	M	1958	
ウッドペッカー	緑の木立ちにウッドペッカーがやって来て次々と木を倒してしまいます。さあ、あなたはどれだけウッドペッカーを生け捕りにできるか？	3,000円	PC-8001(32K) / MICRO-8	B	1959 / 3033	
二次方程式解法テクニック	四則計算から正負の計算、文字式、一次二次方程式にいたるまでの解法を、計算の仕方と基本を重視して展開表示します。式を設定するのはあなたです。	3,000円	PC-8001(32K)	B	1961	
SUPERバルーン	ご存知バルルーンボンバーのオールマシン語によるハイスピード版。ビーム砲とバリヤを駆使して飛行機と風船爆弾を迎撃して下さい。	3,000円	PC-8001(32K)	M	1987	
マウンテン・アタック	落石が頻繁に発生、また至るところに人喰い虎が住んでいます。しかも山頂には怪しげな雷雲が……。果して初登頂なるか？	3,000円	PC-8001	B	3784	
CRAZY DESCENDER	クレイジークレイマーばかりがゲームじゃない！　あなたを狙う二人のオジャママンの攻撃をかわしながら無事地上へおりられるか。	3,000円	PC-8001	B	3785	
BOMBER	HEAD-ONはもう古い。これからはBOMBERゲームの時代です。手に汗にぎる新ゲーム登場!!	3,000円	PC-8001	B	3786	
電気店用 顧客管理システム	1枚のフロッピーには最大630件の顧客が登録可能。顧客の状況をCRT画面上、又はDM用ラベル或は帳表として出力できる。家族状況、商品保有状況、クレジット記録年間月別記録など。	68,000円	PC-8001/FD	ディスク B	3783	
日本語ワードプロセッサー 書記	1ページ最大40字×34行が表示可能！　カナ漢字変換とコード入力、文書はデータとして登録・呼び出しが可能です。印刷と同様に画面表示し、文書の修正・削除も簡単に行なえるなど多彩な文書編集機能を持っています。	9,000円	PC-8801 / MICRO-8	B	3617 / 3021	
SUPERサブマリーン	MICRO—8のグラフィック機能をフルに活用したすばらしいカラー画面です。各種のインジケータを読み取りながら、潜望鏡をのぞいて、敵戦艦20隻を魚雷で撃沈させて下さい。	3,000円	MICRO-8	B	3030	
THE BASEBALL	投手はスロー、スピードボールを選択して投球します。またチェンジアップも可能です。また打球もフライ、テキサス性のヒット，ゴロなど、さまざまです。FM8のすばらしいカラーグラフィック画面で楽しんで下さい。	3,000円	MICRO—8	B	3031	
スペースタッチダウン	ナインドは1,000～300まで、4つの画面に着陸基地・宇宙船・燃料・加速度が表示されています。宇宙船を操縦して無事着陸して下さい。	3,000円	FM-7	M B	3035	
実戦プロ野球ゲーム	ダブルプレーが実現!!セントラルリーグをFM-7でどうぞ	3,800円	FM-7	M B	3022	
実戦グラフィック麻雀ゲーム	ポン・チー・カン、あの感触をマイコンで…。	3,800円	FM-7	M B	3951	
実戦花札ゲーム	さあ来い！勝負だ!?花札の醍醐味を楽しんで。	3,800円	FM-7	M B	3950	
オイチョ・カブゲーム	日本古来のカードゲーム。胴元はもちろんマイコンです。よく考えて勝負して下さい。熱くなって身ぐるみはがされないように。	3,000円	MZ—80B	B	1744	
THE ギャング	大金が眠る豪邸の金庫へたどりつくには、数々の迷路のワープトンネルを利用して突破しなければならない。超一流のギャングである君の行く手を待つのは、大金持ちへの道か、冷たい牢獄か、はたまた大爆発か！	3,000円	PA-7010 パソピア	T B	1960	
SUPER DEFLECTION	／あるいは　の反射板でボールをはねかえしてターゲットに当てるゲーム。①キーはタイプIで、5個のターゲットを1個づつ面変わりで消します。②のキータイプIIは反射板がそのまま残るよりむずかしいゲームです。	2,500円	JR-100	B	1060	
MICRO METEO	地球侵略を狙うメテオ星より隕石群の攻撃がはじまりました。あなたはJR号を操って地球を守ることができるか！	2,500円	JR-100	B	1061	
JR一月面着陸	あなたは月面着陸船のパイロットです。加速度、水平方向速度、降下速度、燃料をよく考えて着陸船を操縦してください。　失敗すると………。	2,500円	JR-100	B	1062	
JR-バルルーンボンバー	風船バクダンを撃ち落とすゲーム。弾丸を連続で撃ち込まないと地面が爆破され動きにくくなる。味方の飛行機を撃つとオソロシイことが………。	3,000円	JR-100	B	1063	

言語　B：ベーシック，M：機械語，HU：HuBASIC，F：FORM-B

テープ名	内容	定価	機種名	言語	コードナンバー	注文数
日本対JR	JR軍団が攻めてきた。勇敢な君は零戦とジェット機を操縦して宿敵JR軍団をやっつけて下さい。	2,800円	JR-100	B	1066	
ジャンケンゲーム	ZX81はガキ大将です。キミはガキ大将と何回戦しますか。	2,000円	ZX81	B	3150	
カードアタック	オーソドックスな神経衰弱ゲームです。さてキミの記憶力は。	2,000円	ZX81	B	3151	
Mini Trek	ワープを駆使して小宇宙に点在するクリンゴンをやっつけよう。	2,300円	ZX81	B	3152	
スペースシップ	反乱をおこしたサーキットボードをさしかえる手に汗にぎるゲーム	2,300円	ZX81	B	3153	
3次元迷路	前後左右へさまよう君。はたしてこの立体迷路から脱出できるかな。	2,300円	ZX81	B	3154	
THE GOLF	あちこちに点在する池と林を計算して、方向と飛距離を決定して下さい。池に落ちたり林に当ったり、コースからはずれると、打数が加算されます。うまくグリーンをとらえられるかな?	2,800円	PC—6001	B	1048	
三次元カーレース	スリル満点。追いつ、追われつ立体カーレース。	3,500円	MZ-700	M	3511	
クラッシュ クリーン	2匹のエイリアンをさけながらあなたの住む町をきれいにしてゆく(白く塗ってゆく)ものです。ワープトンネルを利用して、何面までクリーンに出来るかな?でもエイリアンもワープするから御用心。	2,800円	MZ—1200	B	1897	
三次元カーレース	スリル満点。追いす、追われつ立体カーレース。	3,200円	MZ-1200	M	3466	
THEリーク	ロボットは3つ。アミダでパターンが決まります。○を全てコの字形の中に納めるか、またはリーグが○にぶつかると点になります。	2,800円 3,000円	MZ-1200 PC-8001	B B	3467 3819	
金鉱掘りゲーム	地中海には莫大な金がねむっています。金脈を見つけて一かく千金をねらって下さい。	2,800円 3,000円	MZ-1200 PC-8001	B B	3468 3820	
PC-8801用 実戦プロ野球ゲーム	野球好きのあなた、一度野球チームの監督になってみませんか! もちろんチームは巨人・阪神。江川・中畑・原・篠塚、掛布・岡田が打って走る。	3,800円	PC-8801	B	3619	
THE GOLF	本格的なゴルフシュミレーションゼーム。あちこちに点在する池と林を計算して方向と飛距離を決定。うまくグリーンをとらえられるかな?	3,000円	PC-8001	B	3821	
グラフィックシュミレーション・スターウォー	ブラックホール・戦闘・戦略などアクションがいっぱい。大宇宙戦での指揮官はあなたです。	3,800円	PC-8801	近日発売		
実務トレーニング80B・2000用 Ⓖはグラフィック RAMNo.1 が必要 ※印はMZ-2000には使用不可						
価値判断	マイコンなら入力されたデータにより色付けなしの価値判断が可能。	3,000円	MZ—80B	B	1701	
ローン計算	世はまさにローン一色。マイコンに算出させるのがナウい方法。	2,800円	MZ—80B	B	1704	
多角形の面積計算	もっともポピュラーな多角点測量のデータを計算しデータを求めるプログラム。	3,000円	MZ—80B	B	1705	
多元連立方程式	二元以上、27元までの連立一次方程式を消去法で解答します。	2,800円	MZ—80B	B	1706	
ニュートン法	方程式f(X)=0の近以値解を求めるために、微分を使って算出するソフトウェア。	2,800円	MZ—80B	B	1708	
QSO整理	QSO(交信)記録を手書で整理する時代は、このソフトの登場で終わった!	3,500円	MZ—80B	M	1784	
成績処理(4本組)	1ページ最大20項目×50人×10ページの格納が可能!5段階変換、偏差値変換、生徒番号順一覧、成績順一覧、ヒストグラム、クラス別一覧、クラス別成績順一覧、全クラス総合一覧、個人表など19種類の表が作成可能です。	24,000円	MZ—80B	B	3335	
S-P表作成(2本組)	S—P原表(1.0得点、重みつき誤答分析)、注意係数、平均正答率、信頼性係数、差異係数、標準偏差、S—P表、項目(問題分析)、UL指数、φ係数、クロス表、ソマーズ係数、ヒストグラム、部分S—P表、多肢選択式誤答分析、ヒストグラム、クロス表、集中度係数、平均情報量、相対エントロピー実質選択肢数、偶然得点など。	12,000円	MZ—80B	B	3336	
アンケート集計	1ページ最大250人×10ページのDATA入力が可能、項目(問題)の選択肢数の最大は35、集計結果一覧表、ヒストグラムクロス分析表、クロス分析ヒストグラムがプリントされます。	6,000円	MZ—80B	B	3337	
校内模試(3本組)	クラス数は最大11クラス、1クラス50人です。各科目別ヒストグラム、各科目別得点、偏差値、学年順位一覧、各科目別成績順位一覧表、各クラス別偏差値一覧、クラス別成績順位一覧、学年総合成績順、全クラスの個人成績表出力など。	18,000円	MZ—80B	B	3338	
言語・ソフト for MZ-80B						
FORM-B	Tiny FORTRAN「FORM(フォーム)」がおさめられたソフト。この言語はBASICに比べて高速処理能力があり多目的に使えます。(マニュアル付)	6,000円	MZ—80B※	F	1710	
HU-GBASIC	グラフィック仕様のHU BASICのソフト。話題のHU-BASICでグラフィック機能を強化。表現力バツグンのグラフィックプログラムが組める。	10,000円	MZ—80B※Ⓖ	HUG	1786	
Z80TRACER	Z80の命令をインタープリティブに実行します。1ステップ・レジスタダレプ可能。マシンランゲージモニターはこれ1本ですべてOK。	6,000円	MZ—80B※	M	1793	
プリンター用画面コピー	キャラクターとグラフィックを同時にコピー。40/80文字どちらでもOK。	3,800円	MZ—80B※Ⓖ	B	1794	
HuGキャラクター&メーカー	HuGBASICのPRINT#2命令のためのキャラジェネ作成用プログラム。	3,800円	MZ—80B※Ⓖ	HUG	1795	
実務トレーニング・ソフト for PC-8801						
DM発行システム	DMを多く出される商店や会社用に製作したものですが、家庭の住所録・電話帳がわりにもなります。顧客のランクを登録してランク別にDMをプリントアウトすることもできるため、会員・顧客管理にも使えます。顧客数250以内。	5,000円	PC-8801	B	3600	
ザゴルフ	ハラハラ・ドキドキ!本格的シュミレーションゴルフ	3,000円	PC-8801	B	3620	

テープ名	内容	定価	機種名	言語	コードナンバー	注文数
言語・ソフト編 for MZ-1200、K2E、K/C						
HUBASIC	16桁の倍精度演算、オートラインナンバー、高速演算ルーチンを採用、16進－8進が手軽に操作可能等々。強化BASIC言語のソフト（マニュアル付）	8,000円	MZ－80K/C	HUB	1711	
HU-BASIC COMPILER	高能力言語HU-BASIC用の32Kバイトマシン語コンパイラ。120行 分のコンパイル可能。最敵化機能により実効時が最少ですむ。(マニュアル付)	10,000円	MZ－80K/C	M	1785	
HU-DBASIC	MZ80K/Cユーザ待望のHU-BASICフロッピーバージョのスピーディなローディングで高性能上語HU BACICを思うままに活用できる。(マニュアル付)	16,000円	MZ－80K/C	HU-D	1787	
BASICテキストコンバータ	MZ・80BのBASICプログラムを80K C用に交換します。80Bにあって80K/Cにないステートメントはエラー表示されてLIST UPされます。またキャラクター関係で英小文字リバース英文字は英大文字に自動的にコンバートされます。また1行80文字を越える場合も自動的にエラー表示されますので容易に修正が出来るようになっております。	3,500円	MZ－80K/C	M	1896	

ゲーム・ソフト for MZ-80B

テープ名	内容	定価	コードナンバー	言語	備考	注文数
スーパーバリケード	宇宙空間には、宇宙塵がいっぱい。	3,400円	1086	M	※	
アニマルレッスン	マイコンは動物の知識を増やそうと必死。	2,800円	1718	B		
4人マージャン	マイコン相手の4人マージャンが楽しめる。	6,000円	1799	B	※Ⓖ	
陣取りゲーム	手に汗にぎるスリル満点のゲーム。	2,600円	1720	B	※	
ダービー	出走！ディスプレイ左側から馬が走る。	2,800円	1723	B		
スーパーゴルフ(#2)	本格的なインドア・ゴルフが楽しめる。	4,600円	1798	B	※Ⓖ	
ハングマン	シークレット・ワードを当てるゲームです。	2,800円	1725	B	※	
株式売買ゲーム	5銘柄の相場を50日間取引する。	3,000円	1726	B		
姓名判断	項目別の判断がズラリと表示される。	3,800円	1727	B		
頭の体操 No.1	四つのジャンルをテストします。	3,000円	1728	B		
モールスレッスン	モールス符号のトレーニング用プログラム。	2,800円	1729	B		
チェッカー	相手のコマを飛びこすとコマが取れる"チェッカーゲーム"のマイコン版。	2,800円	1757	B	※	
頭の体操 No.2	No.1の内容を中級クラスまでアップしたプログラム。	3,200円	1758	B		
頭の体操 No.3	No.1、No.2では物足りないという天才向け。	3,400円	1759	B		
TEXASAREA	宇宙船のハッチを開くと、宇宙空間には敵が!?	4,200円	1761	B	Ⓖ	
とりうちゲーム	あなたは迷ハンター？	3,800円	1762	B	Ⓖ	
占星術	ホロスコープを使った占い。	4,600円	1763	B	Ⓖ	
銀河を守れ！	グラフィック画面でスリルあふれる宇宙戦。	3,800円	1764	B	Ⓖ	
医は算術なり	医学生必携のゲーム！	3,600円	1765	B		
キャッチベビー	空からおりてくる赤ちゃんをベビー・ベッドでキャッチ。	3,800円	1766	B	Ⓖ	
宝さがし	砂漠で宝さがしをしよう。	3,800円	1767	B	Ⓖ	
プロファイター No.1	UFO撃墜に全力をあげよ！	2,800円	1768	B	Ⓖ	
プロファイター No.2	ムムム…さすがはNo.2戦闘地区。	3,800円	1769	B	Ⓖ	
アーチェリー	18の的をねらってポイントを競いあうゲーム	3,200円	1771	B	Ⓖ	
プロレーサー	テクニックを駆使してゴールを目ざせ！	3,000円	1772	B	Ⓖ	
エスケープ大作戦	先生の目をぬすんで、学校からエスケープし	3,600円	1773	B	Ⓖ	
うちわでホイホイ	ウチワで紙をあおいでプレーするゲーム。	2,800円	1774	B	Ⓖ	
ICBM迎撃作戦	迎撃ミサイルを操作して防衛作戦を完ぺきに。	3,600円	1775	B	※Ⓖ	
君はターゲット	部屋にかくされているフィルムを発見。	3,200円	1776	B	Ⓖ	
大戦車突破作戦	戦場を進む戦車の操縦士はあなた。	3,200円	1777	B	※Ⓖ	
太陽系一周レース	スケールのデッカイ大レース。	3,400円	1778	B	Ⓖ	

テープ名	内容	定価	コードナンバー	言語	備考	注文数
吸血鬼撲滅作戦	十字架とニンニクを使って撲滅作戦。	3,800円	1779	B	※Ⓖ	
ターゲットライセンス	プロの射手は道はけわしい。	3,000円	1780	B	Ⓖ	
蚊取り線香は死の臭い	おそってくる蚊をやっつけろ。	3,000円	1781	B	Ⓖ	
スキー・ゲーム	直滑降、回転10のスキーコース。	3,600円	1778	B	※Ⓖ	
海賊ゲーム	海賊船に乗りうつられますゾ！	3,800円	1789	B	Ⓖ	
鮫うちゲーム	危険なサメ打ちのプロハンター。	3,800円	1791	B	Ⓖ	
ビンゴゲーム	先につぶした列が5列できた方が勝ち。	3,800円	1792	B		
オカルトハウス	捕ったり部屋に閉じ込められないように注意。	3,600円	1080	B	※Ⓖ	
プレイボーイゲーム	箱入り娘を外へつれだすゲームです。	3,800円	1081	B	Ⓖ	
水戸黄門ゲーム	助さん、格さんを動かして、やっつけてください。	3,600円	1082	B	※Ⓖ	
神経衰弱	さあ、あなたの記憶力は？	3,400円	1084	B	Ⓖ	
タコタコあがるな	穴から飛び出したタコを網で捕えるゲーム。	3,600円	1085	B	Ⓖ	
スーパーブロックくずし	3種類のブロックが楽しめる。	3,400円	1087	M	※	
駆逐艦撃沈ゲーム	魚雷は20発しかないので正確に。	3,600円	1088	B	Ⓖ	
アステロイドベルト	小惑星を破壊せよ！	4,000円	3320	M	※Ⓖ	
ミサイルコマンド	"飛来するミサイル群"から都市を守れ！	4,000円	3321	M	※Ⓖ	
ポラリス	海中から敵飛行部隊を全滅せよ！	4,000円	3322	M	※Ⓖ	
スペースウォーズ	打ちまくれ！ 2人でも遊べます。	3,000円	3323	M	※Ⓖ	
スネーキー	カエルを喰べて、どんどん長くなります。	3,000円	3324	B	Ⓖ	
エイリアン	迫るエイリアンを巧妙に避けろ！	3,000円	3325	M	※	
スペースビー	ハチの巣を狙え。ギャラクシアンゲーム。	3,000円	3326	M	※	
パックマンファイト	パワーエサを食べたらそれっ！逆襲だ。	3,000円	3327	M		
バクテリア	突然出現した猛毒バクテリア！	3,000円	3328	M	※Ⓖ	
恐怖のエイリアン	攻撃をキミはどこまで防ぎきれるか!?	3,000円	3329	B		
クレージーダンゴ	ダンゴを求めて死地を走りまわる。	3,000円	3330	M	※	
ドッグファイト	華麗なる空中戦、敵機に弾丸を20発打ちこめ。	3,000円	3331	M	※	
レーダーパックマン	迷路ゲームの決定版。	3,000円	3332	M	※Ⓖ	
アステロイドウォーズ	なんと80Bが言葉をしゃべる。	3,000円	3332	B	※Ⓖ	
バグファイター	恐怖のバグと、ハンマー片手に闘え！	3,000円	3334	M	※Ⓖ	
バイオリズム	あなたのバイオリズムが一目でわかります。	3,500円	3341	B	Ⓖ	
ミステリーハウス	アドベンチャーゲームの決定版。	5,200円	3343	M	※Ⓖ	

ゲーム・実務トレーニング・ソフト for MICRO-8
(FM-7にも使用可)

※印はMZ－2000には使用不可　Ⓖはグラフィック RAM No.1が必要

テープ名	内容	定価	コードナンバー	言語	備考	注文数
ダービー	単勝馬券で5名まで参加OK！	3,000円	1900	B		
月面着陸	あなたは月面着陸船の操縦士。	3,000円	1902	B		
アルデバラン ＃1	BASICゲームの古典的名作。	3,000円	1903	B		
スタートレック	マイコンゲームの古典的名作。	3,600円	1904	B		
アニマルレッスン	マイコンを動物学者にしてしまおう。	3,000円	1905	B		
頭の体操 No.1	四つのジャンルをテストします。	3,200円	1906	B		
ニュートン法	方程式f(x)=0の近似値解を求める。	3,000円	1907	B		
多角形の面積計算	測量用の実用ソフトウエア。	3,000円	1908	B		
多元連立方程式	二元以上、27元までの一次方程式を計算	3,000円	1909	B		
表集計	X、Yの表計を計算してくれます。	3,600円	1910	B		
英会話レッスン	マイコンがランダムに出題してくれる設問	3,000円	1912	B		
価値判断	色付けなしの価値判断が可能。	3,400円	1913	B		

ゲーム・ソフト for PC-8001

テープ名	内容	定価	コードナンバー	言語	備考	注文数
レーダーサーチ	レーダーを使って敵を攻撃します。	3,000円	1914	BM	(32K)	
ばぐごん	ばぐごんにみつからないように。	3,000円	1915	BM	(32K)	
シリウスF	君は地球へ帰ることができるか。	3,000円	1916	BM	(32K)	
エアーライフル	11発の弾丸で君は何点かせげるかな？	3,000円	1917	BM	(32K)	
キングタイガーIII PART1	"キングタイガー"迎え撃つシャーマン戦車。	3,000円	1918	BM	(32K)	
バトルバルカン	君は宇宙パトロールの隊員だ。	3,000円	1919	BM	(32K)	
ビッグアステロイド	無事に地球へ帰還ができるか.	3,000円	1920	BM	(32K)	
ブラックホール	新兵器プロトン砲を使いホワイトホールへ脱出しろ。	3,000円	1921	BM	(32K)	
戦艦大和	特命を受けた戦艦大和は沖縄に向け出発する。	3,000円	1922	BM	(32K)	
ドキドキすいか割り	すいか割りを楽しんでいる。ところが…。	3,000円	1923	B	(32K)	
アスロック	海上には戦艦、機雷が雨のように降ってくる.	3,000円	1924	BM	(32K)	
ワイルドスワット	暴走族絶滅の為、今日もパトロール。	3,000円	1925	BM	(32K)	
プラネットバルカン	宇宙暦2999年。遂に惑星間戦争に突入してしまった。	3,000円	1926	BM	(32K)	
ギャラクティカ 1	スクリューカノン砲で敵を破壊せよ。	3,000円	1927	BM	(32K)	
ギャラクティカ 2	反乱軍は侵入者に対して無差別攻撃を加えてくる。	3,000円	1928	BM	(32K)	
ギャラクティカ 3	太陽の重力変化の為絶滅の危機。	3,000円	1929	BM	(32K)	
バトルファイヤー	せまりくる敵大船団。	3,000円	1930	BM	(32K)	
CRTチェイサー PART1	10ヶのチェックポイントを周らねばなりません。	3,000円	1931	BM	(32K)	
CRTチェイサー PART2	宇宙機雷と目に見えない境界線に注意。	3,000円	1932	BM	(32K)	
CRTチェイサー PART3	探査装置をかついて、埋蔵金を探しあてて下さい。	3,000円	1933	BM	(32K)	
CRTチェイサー PART4	あなたは地底の迷路に迷いこんでしまいました。	3,000円	1934	BM	(32K)	
マリンどんべえだあ PART1	汚染の海を清掃中。回収のゴミの中に不発弾も交っている。	3,000円	1935	BM	(32K)	
マリンどんべえだあ PART2	どんべえII世号は外洋掃除の任務につきました。	3,000円	1936	BM	(32K)	
スカイどんべえだあ PART1	大気汚染調査隊は今日も調査に出かける。	3,000円	1937	BM	(32K)	
スカイどんべえだあ PART2	2機で回ることになったスカイどんべえ。	3,000円	1938	BM	(32K)	
スペースどんべえだあ PART1	エイリアンは惑星の陰にかくれています。	3,000円	1939	BM	(32K)	
スペースどんべえだあ PART2	奴らのミサイルはなかなか強力です。	3,000円	1940	BM	(32K)	
タマツキゲーム	角度と強さが決ったら、さあGO!!	3,000円	1941	B		
スペースランディング	宇宙ステーションマザーIに着艦しなければならない。	3,000円	1942	BM	(32K)	
フューチャー	ブラックホールに吸いこまれたら生きて帰れない。	3,000円	1943	BM	(32K)	
侵略ゲーム	直射、斜撃砲を使って撃ち抜き防戦。	3,000円	1989	B	(32K)	
大脱走ゲーム	ランクも3段階と充実。	3,000円	1990	B	(32K)	
脱獄ゲーム	何人を脱獄させることができるか！	3,000円	1991	B	(32K)	
七並べトランプ	トランプゲームコンピュータ相手に楽しもう。	3,000円	1993	B	(32K)	
迷探偵ゲーム	通り過ぎる犯人をつかまえよう。	3,000円	1994	B	(32K)	
ファイアー・インフェルノ	ビルが大火事。さあたいへん。	3,000円	1995	B	(32K)	
スクランブルチェイサー	追いつ追われつ、2人で楽しみ3倍増。	3,000円	3763	BM	(32K)	

ゲーム・実務トレーニング・ソフト forパソピアPA-7010

テープ名	内容	定価	コードナンバー	言語	備考	注文数
SS計算	平均点、順位、偏差値などを処理。	2,800円	1200	B		
金種計算	経理課の悩みの種もこれで楽々！	2,800円	1201	B		
アルデバラン ＃1	スタートレックをしのぐBASICゲームの	3,600円	1202	B		
スーパースタートレック	マイコンゲームの古典的名作。	3,800円	1203	B		
ビンゴ 25	たて、よこ、ななめ、早く5列に並べた方が勝ち。	3,600円	1204	B		
アニマルレッスン	マイコンは動物の知識を増やそうと必死	3,000円	1205	B		
頭の体操 No.1	四つのジャンルをテストします。	3,000円	1206	B		
頭の体操 No.2	頭の体操No.1を中級クラスまでアップ。	3,200円	1207	B		
頭の体操 No.3	これで高得点が取れたら尊敬します。	3,400円	1208	B		
キーボードレッスン	正確なインプットをするためのソフトウェア。	3,200円	1209	B		
ハングマン	マイコンが指定するシークレット・ワード。	2,800円	1210	B		
殿様ゲーム	あなたはエゾの国の大将。	2,800円	1211	B		
株式売買ゲーム	5銘柄の相場を50日間取引する。	2,800円	1212	B		
チェッカー	相手のコマを飛びこすとコマが取れる"チェッカー・ゲーム"	3,000円	1213	B		
英会話レッスン（上級）	英会話でよく使う表現の基本編。	3,400円	1214	B		
英会話レッスン（中級）	英会話でよく使う表現の応用編。	3,200円	1215	B		
ベーシック・レッスン No.1	コマンドの説明用プログラム。(入門編)	3,000円	1216	B		
ベーシック・レッスン No.2	コマンドの説明用プログラム。(基礎編)	3,000円	1217	B		
ベーシック・レッスン No.3	コマンドの説明用プログラム。(応用編)	3,000円	1218	B		
ニュートン法	方程式f(x)＝0の方似値解を求める。	2,800円	1219	B		
多元連立方程式	二次以上、27元までの連立一次方程式を解答。	2,800円	1220	B		
月面着陸	距離と高度を見ながら着陸。	3,600円	1221	B		
バリケード	星を壁にぶつからないように。	3,000円	1223	B		
ブロックくずし	ボールは全部で7個。さて何点とることができるか？	3,200円	1224	B		
陣取りゲーム	相手に陣地をとられないように。	3,000円	1225	B		
ハノイの塔	並んだ円盤を崩さず移動しましょう。	3,000円	1227	B		
占星術	相性、恋愛運等も教えてくれる。	4,200円	1230	B		
医は算術なり	医学生必携のゲーム？	3,400円	1231	B		
価値判断	コンピューター的使い方決定版！	3,000円	1232	B		

テープ名	内容	定価	コードナンバー	言語	備考	注文数
英単語レッスン(初級)	設問にキーボードで解答。	2,800円	1233	B		
英単語レッスン(中1用)	設問にキーボードで解答。	3,000円	1234	B		
英単語レッスン(中2用)	設問にキーボードで解答。	3,000円	1235	B		
英単語レッスン(中3用)	設問にキーボードで解答。	3,000円	1236	B		
モールスレッスン	パソピアでモールス練習。	3,000円	1238	B		
ローン計算	簡単なデータ入力でマイコンに算出。	3,000円	1239	B		
測量計算	三点の座標を入れて面積を計算する。	2,800円	1240	B		
多角形の面積計算	多角点測量のデータを計算。	2,800円	1241	B		
2001年宇宙の旅 PART1	HAL 9000の反乱を、どう止めるか！	3,500円	1242	B		
2001年宇宙の旅 PART2	スタートゲートを通り過ぎた。そこにはコクセキヒが……。	3,500円	1243	B		
ウォーク・ワン	13丁目の聖子ちゃんの家に行くが……。	3,500円	1244	B		
コンピューター BuG9000	HAL-9000を作ろうとしていたが……。	3,500円	1245	B		
ザ・ゴルフ	フルカラー・ゴルフをお楽しみ下さい。	3,000円	1246	B		

ゲーム・ソフト for PC-6001

テープ名	内容	定価	コードナンバー	言語	備考	注文数
ビンゴゲーム	コンピューター相手にマス目つぶし。	3,200円	1000	BM		
アルデバラン #1	SFストーリー・ゲームのPC-6001版。	3,200円	1001	BM		
チェッカー	相手のコマを飛びこすとコマが取れるチェッカーゲーム。	3,000円	1003	BM		
株式売買ゲーム	相場師としてのウデマエをためすいいチャンス！	3,000円	1006	BM		
スペースシューティング	UFOをレーダースクリーンでとらえて撃破。ジョイスチック可。	3,400円	1007	BM		
医は算術なり	ハタで見るほどお医者さんも楽ではないですね。	3,000円	1009	BM		
大戦車突破作戦	マシン語によるタンクゲーム。ジョイスティック可。	3,800円	1010	BM		
バルチック艦隊	バルチック艦隊接近！砲台を死守せよ。ジョイスティック可。	3,200円	1042	BM		
アステロイドエクスプレス	飛び交う小惑星をかいくぐり、どこまで行けるかな。ジョイスティック可。	3,400円	1043	BM		
プロレーサー	ゲームセンターでおなじみのカーレーサー。ジョイスティック可。	3,200円	1044	BM		
タイタンファイター	タイタンの侵入者から地球を守れ！地球の運命は？	3,200円	1045	BM		
スペースビー	宇宙バチが襲ってきた、ハチの巣を狙え!!	3,000円	1047	B		
FX空中戦	バリヤーの中で何機コアファイターを破壊できるか。ジョイスティック可。	3,200円	1046	BM		
ブロックくずし	ブロックくずしの超高速版。パドルの大きさと、球速は調整可。	3,000円	1025	BM		
U・F・Oくずし	ブロックに囲まれたUFOを撃墜して下さい。	3,000円	1026	BM		
五目ならべ	パピコン相手に五目ならべはいかが。	3,000円	1027	BM		
ミステリーハウス	アドベンチャーゲームの決定版。	3,800円	1024	B		
PC-6001用 百人一首	学習とゲームを両立したソフトテープ登場。	3,000円	1049	B	(32K)	
クラッシュ・ラリー	何個のポイントマークをとることができるか。	2,800円	3102	BM		
街道レーサー	並み居る車をかわし、快音を轟かせて走る街道レーサー。	2,800円	3103	BM		
ブラックボックス	ボールをレーザービームでうまく発見できるか。	2,500円	3104	BM		
UFOゲーム	敵円盤は高性能ランダム走行メカを揃えている。	2,800円	3105	BM		
スピット・ファイヤー	撃墜王になることができるか。	2,800円	3106	BM		
ギャラクティカ ①	ゴモラがミサイル攻撃をしかけてきた。	2,800円	3107	BM		
モナコ・グランプリ	デッドヒート。スリップゾーンも待ち受ける。	2'800円	3108	BM		
ディープ・スキャン	水面下は敵潜水艦ばかり。	2,800円	3109	BM		
ばくごん	敵は"赤べえ"だ。エネルギーカプセルを次々食べて突激！	2,800円	3110	BM	(32K)	
バグ・パニック	奇怪な虫が出没。強暴な土人が襲ってくる。	2,800円	3111	BM		
エイリアン	君の武器は如意棒と、ずばぬけたジンプ力だけ。	2,800円	3112	BM	(32K)	
フォボス	火星衛星フォボスに隠された4つの異次元の穴。	2,800円	3113	BM		
ビルディングホッパー	駆け上がる先からゴロゴロ転ってくる障害物。	2,800円	3114	BM		
キングタイガー ③	シャーマン戦車と市街戦が続く。	2,800円	3115	BM	(32K)	
クレイジーニュートン	リンゴが1個また1個……。なんとその中に爆弾が混入！	2,800円	3116	BM	(32K)	
スネーキー	栄養満点のカエルを食べて、どんどん長くなります。	3,000円	3117	BM		
ヘッドオン	ご存知ヘッドオンわき見をするな！	3,000円	3118	BM		
エクセリアン	前後左右さら攻撃して来る敵を狙え。	3,000円	3119	BM		
ディグバグ	迫り来るエイリアンを岩石落して逆襲だ。	3,000円	3120	BM		
金種計算	経理課のお手伝い。200名までOK。	2,800円	1801	B		
ローン計算	銀行の金利やローンの返済を計算。	2,800円	1802	B		
SS計算	試験の成績計算。	2,800円	1803	B		
ニュートン法	f(x)＝0の解を求める。	2,800円	1804	B		
多元連立方程式	二次以上の連立方程式を解く。	2,800円	1805	B		
測量計算	三点の座標を入れる事により面積を計算。	2,800円	1806	B		
数学レッスン	関数を入れ微分・積分の計算。	2,800円	1807	B		
アルファベットのおけいこ	画面に出たらひらがなアルファベットのキーをさがす。	2,800円	1810	B		
キーボードレッスン	キーボードを早く打つ為のレッスン。	2,800円	1811	B		
9×9レッスン	コンピュータが小学生に九・九を教えます。	2,800円	1812	B		
たし算引き算	マイコンの幼児教育はまずたし算・ひき算から。	2,800円	1813	B		
デジタル計算	マイコンが目覚し時計に早かわり。	2,800円	1818	B		
世界時計	マイコンが世界の時刻を知らせます。	2,800円	1819	B		
マイコン・キューブ	マイコンでルービック・キューブに挑戦。	2,800円	1821	B		
殿様ゲーム	エゾの国の経営はあなたの采配で。	2,800円	1822	B		
ハングマン	死刑囚を救うのは、あなたの頭しだい。	2,800円	1823	B		
陣取りゲーム	迅速で敏感な判断を要するゲーム。	2,800円	1824	B		
ベーシック・レッスン No. 1	PC-6001のコマンドの説明用プログラム。(基礎編)	2,800円	1829	B		
ベーシック・レッスン No.2	PC-6001のコマンドの説明用プログラム。(応用編)	2,800円	1830	B		
英単語レッスン(中3用)	中学3年で学習する英単語。	2,800円	1835	B		
英単語レッスン(初心者用)	英単語でよく使用されるもののレッスン。	2,800円	1836	B		
バリケード	キーボード操作で宇宙塵をつかまえろ。	2,800円	1840	B		
頭の体操 No.1	四つのジャンルをテストしよう。	2,800円	1837	B		
頭の体操 No.2	頭の体操No.1を中級クラスまでアップ。	2,800円	1838	B		
頭の体操 No.3	これで高得点が取れたら尊敬します。	2,800円	1839	B		

ゲーム・ソフト for MZ-1200
(K2E・K・K2・Cにも使用できます)　✲印はMZ－700にも使用可

テープ名	内容	定価	コードナンバー	言語	備考	注文数
ボーリング	キー操作でストライクを出して下さい。	2,500円	1841	B	✲	
スロットマシン	同じ絵がいくつ並ぶかな？	2,500円	1842	B		
スタートレック	マイコンの古典的古典的名作。	2,800円	1843	B		
ヤシの実落し	土人が5人でヤシの実を確保。	2,500円	1844	B		
価値判断	コンピュータ的使い方決定版。	3,000円	1845	B		

テープ名	内容	定価	コードナンバー	言語	備考	注文数
金種計算	経理課の悩みの種もこれで楽々。	2,500円	1846	B	✲	
パチンコ	チューリップに入れば高得点。	3,000円	1847	B		
ベースボール	マイコンチームは強敵チームだ。	2,500円	1848	B		
殿様ゲーム	あなたはエゾの国の大将。	2,500円	1849	B	✲	
バリケード	壁にぶつからないように確保してください。	2,800円	1850	B		
水泳	1から5コースまでの水泳・自由競技。	2,000円	1851	B	✲	
ブロッククズシ	ボールは全部で7個。さて何点とることができるか?	2,600円	1853	B		
アニマルレッスン	マイコンを動物学者にしてしまおう。	2,600円	1854	B	✲	
マージャン	メンツがなくてもマイコン相手にマージャンが楽しめるゾ!	3,000円	1855	B	✲	
陣取りゲーム	相手に陣地をとられないよう敏感な判断が必要。	3,000円	1856	B	✲	
さるも木から落ちる	4本の木の間をサルが飛びかいます。	3,000円	1857	B		
チェッカー	相手のコマを飛びこすとコマが取れる。	2,800円	1858	B		
ポーカー	一人で楽しむポーカー。	3,000円	1859	B	✲	
雀球	マージャン・パチンコのマイコン版。	3,000円	1860	B		
野球拳	ちょっと色っぽい、おなじみのよいのよい。	2,800円	1861	B	✲	
ブラックジャック	カードの合計が21に近づくようにします。	3,000円	1862	B	✲	
ダービー	あなたは何ワクに賭けますか?	2,800円	1863	B		
パックパック	パワーえさを食べたら逆襲だ!	3,400円	1893	M		
スーパーゴルフ	ドンカーに落ちないように注意して…。	3,800円	1865	B	48K	
ハングマン	シークレットワードを当ててください。	2,800円	1866	B		
D-DAY	史上最大の作戦を阻止しよう。	3,000円	1867	B		
アルデバラン #1	BASICゲームの決定版。	3,000円	1868	B		
アルデバラン #2	アントスへの旅立ち。	3,500円	1869	B	48K	
戦国軍団	おかしなおかしな戦争ゲーム。	3,000円	1870	M	✲	
月面着陸	距離と高度を見ながら月へ着陸して下さい。	2,800円	1871	B		
カンニング大作戦	カンニングはスリル満点!?	3,000円	1872	B	48K	
スクランブル	フライトシュミレーションとUFOを撃墜するゲーム。	3,000円	1873	B		
モンタージュ	モンタージュの顔の人を探して下さい。	2,500円	1874	B	✲	
株式売買ゲーム	5銘柄と相場を50日間取引する。	3,000円	1890	B		
アステロイド	迫りくる敵をやっつけろ!	3,000円	1891	F		
QSO整理	QSO(交信)記録をマイコンで整理。	3,500円	1877	M		
RAM TEST	メモリーチェックプログラム。	2,500円	1878	M		
NEWテンキー & ファンクションキー	グラフィックキーが10個のファンクションキーとテンキーに早変わり。	3,800円	1879	M		
MZ-TONE	MZ-80が電子オルガンになり作曲もできる。	3,000円	1880	F		
在庫管理	総合的な在庫管理を行なう。日常の商品入出庫を管理。	3,000円	1881	B	✲	
多角形の面積計算	多角点測量のデータ計算。	3,000円	1882	B	✲	
ローン計算	簡単なデータ入力で算出可能。	2,800円	1883	B	✲	
多元連立方程式	二元以上,27元までの連立一次方程式を解答。	2,800円	1884	B		
表集計	表作成に必要な数値を入力、X・Yの表計を計算。	2,800円	1885	B		
SS計算	平均点、順位、偏差値などを処理。	2,800円	1886	B		

テープ名	内容	定価	コードナンバー	言語	備考	注文数
ニュートン法	方程式f(x)=0の近似解を求める。	2,800円	1877	B		
Z-80/TRACER	マシン語の開発になくてはならないツール。	6,000円	1888	M		
顧客管理	顧客数250、項目数は1~8まで可能。自店の使用目的に合わせて利用。	4,000円	1889	M		
バレーボール	サーブ、レシーブ何でもできる立体バレーボール。	4,200円	1756	HuB	48K	
スネーキー	栄養満点のカエルを食べて、どんどん長くなりますか?	3,000円	3450	M		
エイリアン	君の武器は如意棒とずばぬけたジャンプ力だけ。	3,000円	3451	M		
バグファイヤー	恐怖のバグと、ハンマー片手に闘え。	3,000円	3452	M		
パックマン	パワーえさを食べたら逆襲だ。	3,000円	3453	M	48K	
恐怖のエイリアン	攻撃を君はどこまで防ぎきれるか!?	3,000円	3454	M		
クレージーダンゴ	これはもう狂気の世界だ!	3,000円	3455	M		
ポラリス	海中から敵飛行部隊をせん滅せよ!!	3,000円	3456	M		
レーダーパックマン	迷路ゲームの決定版。	3,000円	3457	M		
ドッグファイト	華麗なる空中戦。	3,000円	3458	M		
UFOくずし	ブロックに囲まれたUFOを撃墜して下さい。	3,000円	3459	M		
スターファイヤー	前方射程内に敵機発見!	3,000円	3460	M		
ダンゴ	ダンゴーをめざせ!敵の間げきをぬって通過せよ!	3,000円	3461	M		
DONT'ストップ	道を誤るな!尾行を振り切れ!	3,000円	3462	M		
ニューラリーX	追っては逃げ、逃げては追って。	3,000円	3463	M		
アルカディア、アルカディウム	いん石をさけながら愛機を操縦。	3,000円	3464	M		
宝ビル アドベンチャー	宝ビルは迷路。立ち入り禁止もあります。	2,800円	3465	B	(48K)	

ゲーム・ソフト for MZ-2000

テープ名	内容	定価	コードナンバー	言語	備考	注文数
ポラリス	海中から敵飛行部隊を全滅せよ!	4,000円	3400	M	Ⓖ	
ミサイル ディフェンダー	敵機を探せ!ミサイル発射!	3,000円	3401	M	Ⓖ	
ドッグファイト	華麗なる空中戦!	3,000円	3402	M	Ⓖ	
エイリアン	迫るエイリアンを巧妙に避けろ。	3,000円	3403	M		
アマテニス	すばやい動き、ディスプレイ・テニス。	3,000円	3404	M	Ⓖ	
スペースピー	ハチの巣を狙え。	3,000円	3405	M		
4人マージャン	メンツがいなくてもマイコンで4人マージャンができる。	6,000円	3409	M		
バイオリズム	あなたのバイオリズムが一目でわかります。	3,500円	3410	B	Ⓖ	
スタートレック	スタートレックのMZ-2000版。	2,800円	3413	B		
パチンコ	チューリップに入れて高得点。	3,000円	3414	B		
ダービー	出走!ディスプレイ左側から馬が走る。	2,800円	3415	B		
スーパーゴルフ	本格的なインドア・ゴルフが楽しめる。	3,800円	3416	B	Ⓖ	
大戦車突破作戦	戦場を進む戦車の操縦士はあなた。	3,200円	3417	B	Ⓖ	
パックマンファイト	パワーえさを食べたらそれっ逆襲だ!	3,000円	3418	M		
スネーキー	カエルを食べてどんどん長くなります。	3,000円	3419	B	Ⓖ	
フォボス	火星の衛星フォボスの影に4つの異次元の穴	3,800円	3420	BM	Ⓖ	
出世ゲーム	人生は波瀾万丈ですぞ!!	4,200円	3421	B		
グラフィックポーカー	スーパーコンピュータ対さすらいギャンブラーの対マン勝負。	3,800円	3425	B	Ⓖ	
ミステリーハウス	アドベンチャーゲームの決定版。	5,200円	3346	M	Ⓖ	

✲印はMZ－700に使用可　Ⓖはグラフィック RAM No.1が必要

マイコン別冊
PC-8001マシン語入門II
塚越一雄 著

昭和58年3月30日発行

定価 1,300円（送料250円）

発 行 人　平山秀雄
発 行 所　(株)電波新聞社
郵便番号141 東京都品川区東五反田1－11－15
電話(03) 445－6111(大代) 振替東京5－51961
印 刷 所　大日本印刷(株)
製 本 所　(株)堅 省 堂

変わるね、PCとのつきあい。

フロッピーディスク ￥39,800

フロッピーが僕でも使えるようになった。マイクロディスク新発売！

フロッピー

カセットデッキでゲームをロードする時のあのイライラした気持ち…「こんな時にディスクドライブがあったら」…マイコンフリークなら誰でも経験したことがあるに違いありません。しかし、従来のディスクドライブはゲーム、ホビーに使うにはあまりにも高価で気軽に使うにはほど遠い存在でした。5万円を切るディスクドライブなんて夢のまた夢…もうあきらめ半分でカセットのボタンを押したものです。

しかし、夢は駆け足で現実になりました。マイクロディスクFC-80、気楽な、気楽なディスクドライブの登場です。

■2.7インチのミニサイズフロッピー
2.7インチのサイズはもちろん世界ではじめて。しかもミニサイズながら16KBの容量。(両面)

■ディスクは何枚にわたっても自動的に継続
オートマージ、自動継続機能つきなので、どんな長いプログラムでもセーブ、ロード可能。

■マシン語のゲームもOK
マシン語のゲームは、従来のものではDISK BASICのシステムエリアと重なりミニディスクへのマシン語のロード、セーブが不可能でしたが、マイクロディスクではマシン語もBASICと同様にロード、セーブできます。

■ユーザー本位のテキストエリア
システムを拡張ROMエリアの7800H～7FFFHに持っているので、ユーザーはPC-8001のテキストエリアをフルに利用できます。

■BASICは中間言語でセーブ
プログラムはBASICのままでなく、中間言語の形でセーブするので、BASICのままの場合と比べ約25%記憶容量が節約できます。たとえば、ディスク1枚半で32KのBASICプログラムを完全に記憶できます。

■周辺機器との接続も可能
マイクロフロッピーの使用しているI/OアドレスはPC本体および各周辺機器もまったく使用していません。したがってT字型のバスコネクターを使用すればそのまま周辺機器の接続ができます。

■ハードコピーROMも同時に使用可能
システムROMは拡張ROMエリアの7800H～7FFFHを使用しているので、現在ハードコピーROMを使用している方も同時に使用することができます。

■スピーディーなセーブ、ロード
プログラム(BASIC、マシン語)のロードは8秒でOK(片面時)。もちろんセーブ、ロード時のエラーは完全追放。

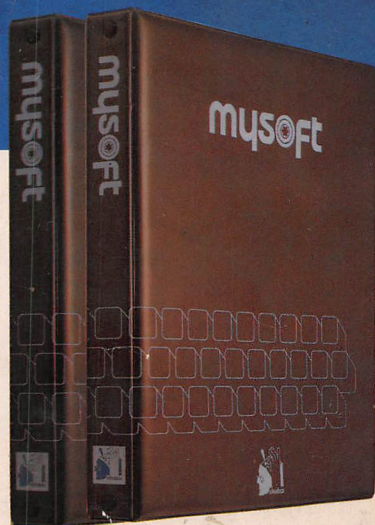
■驚異の低価格
常識を破る￥39,800(システムROMケーブル付)の低価格。まさにカセットに代わる新しいタイプの記憶媒体です。

●データ転送速度25,000bps ●高信頼性、高性能のDDモーター内蔵 ●無改造で接続可能(専用ケーブル、ROMシステムソフト付) ●外形寸法：130W×81H×260D(m/m)

★マークII用アダプター新発売!!
その他PCシリーズ用近日発売!!

お買求めは全国有名マイコンショップまたはCSKメイリングサービセスを御利用下さい。

CSKソフトウェアプロダクツ

〒160 東京都新宿区歌舞伎町1-5-4 第6荒井ビル ☎03-207-3041

ideabox
マイコン別冊 PC8001マシン語入門Ⅱ
昭和五十六年三月三十日発行
mysoft
D2 Mini-Floppy Disk
name 在庫管理
version 2.1
date H101
by 株式会社東海クリエイト
mysoft
マイソフト在庫管理は
オフィスを選ばない
マイソフトの販売管理システムシリーズ，在庫管理は，取扱い商品3,000種，仕入先450社と他に類を見ぬ大容量処理能力を実現。さらに，あらゆる業態にフィットする柔軟性，正確な現状把握，商機をのがさぬ適正在庫把握を実現した各種帳票類，だれにも使える簡易性など在庫管理の究極を追求しました。
mysoft
マイソフト在庫管理 PC-8000シリーズ用 ディスク版 97,000円
総販売元 関東電子株式会社
開発元 株式会社 東海クリエイト
営業部 〒101 東京都千代田区神田須田町1-5(KSビル8F) ☎03(251)3291
大阪支店 ☎06(632)0207
町田営業所 ☎0427(28)8882
多摩営業所 ☎0424(65)8188
群馬営業所 ☎0270(23)2301
仙台営業所 ☎0222(33)0257
名古屋営業所 ☎052(263)1693
京都営業所 ☎075(343)0995
福岡営業所 ☎092(474)5777
広島営業所 ☎082(227)5536
関東バイトショップ ☎03(253)5264
バイトショップコーヨー ☎03(255)6504
バイトショップアキバ ☎03(251)2329
阿佐ヶ谷バイトショップ ☎03(314)3272
伊勢崎バイトショップ ☎0270(23)2302
仙台バイトショップ ☎0222(33)0256
名古屋バイトショップ ☎052(263)1629
大阪バイトショップ ☎06(644)1548
福岡バイトショップ ☎092(474)5778
岡谷バイトショップ ☎02662(3)1075
mysoft
電波新聞社 〒141 東京都品川区東五反田1-11-15 電話03(445)6111 定価1,300円 雑誌08370－3